Sage X3 Web Services v5.0

Formation Web Services

Sommaire

Formation Web Services Sage X3

1.1 Généralités sur les services web

Qu'est-ce qu'un service web ?

- Un service web est une technologie qui permet à une application de s'ouvrir vers l'extérieur pour proposer des fonctionnalités à d'autres applications, en dialoguant à distance via Internet. Ceci, indépendamment des plates-formes et des langages sur lesquelles elles reposent.

- Chaque service web propose un ensemble de fonctionnalités (leur savoir-faire) appelé méthodes.

- Les services Web s'appuient sur un ensemble de protocoles (**SOAP** : **S**ingle **O**bject **A**ccess **P**rotocol) standardisant les échanges.

- La technologie des Services Web englobe de nombreux concepts et tend à s'imposer comme le nouveau standard en terme d'intégration et d'échanges B2B et B2C.

1.2 Les services web Sage X3

- Les services web adonix reposent sur des standards définis et non sur un protocole propriétaire. Ils permettent la communication entre X3 et d'autres applications.

- Ces applications peuvent couvrir de nombreux domaines :
  - Site Internet E-Commerce B2B/B2C
  - Application métier
  - ...

- Chacune des ces applications ont des points communs :
  - Un fort besoin d'interactivité avec l'ERP :
    - ✓ Mise à disposition d'informations « up to date »
    - ✓ Mise à jour de l'ERP en temps réel
  - Une sécurité accrue
    - ✓ Pas de fichier ASCII transitant sur des disques durs

1.3 Méthodes des services web ADONIX

- **X3 propose un web service qui regroupe les méthodes suivantes :**

1. **Méthodes des Objets** : permet d'accéder à X3 via les objets «métier» du progiciel.

2. **Méthodes des Sous Programmes** : permet l'exécution de programmes développés dans X3.

3. **Méthodes des listes** : permet la consultation d'informations basées sur le paramétrage des objets «métier», liste gauche des objets sage X3.

1.3.1 Le Service Web Objet / Description

- **Les méthodes des Objets permettent d'accéder aux objets « métier » du progiciel. Tels que :**
  - ✓ Commandes de ventes
  - ✓ Articles
  - ✓ Clients
  - ✓ … Tout objet X3 est éligible

  – Seuls les objets liés aux cycles de vente sont certifiés par Sage (commandes, articles, clients)

  – Les méthodes liés aux objets peuvent être appliquées à n'importe quel objet, elles restent figées même si l'objet évolue (patch, changement de version …)

1.3.1 Le Service Web Objet / Méthodes

1. **Lecture d'enregistrements : Read(contexte, objet, clé)**
   - ✓ On demande à X3 de lire un enregistrement, on donne la ou les valeurs de clé. Cela revient à consulter une fiche dans une gestion objet, l'enregistrement n'est cependant pas verrouillé.
   - ✓ X3 nous répond en nous donnant les valeurs de l'ensemble des champs des écrans de l'objet concerné.

2. **Création d'enregistrements : save(objet,flux)**
   - ✓ On demande à X3 de créer un enregistrement, on lui transmet tous les champs nécessaires ainsi que ceux rendus obligatoires dans les écrans. L'ensemble des contrôles applicatifs sont déclenchés. C'est l'équivalent du bouton créer de la gestion objet.
   - ✓ X3 nous répond en nous donnant la valeurs de tous les champs des écrans et les messages d'erreurs éventuels, on sait si la création a été effectuée ou non.

1.3.1 Le Service Web Objet / Méthodes

3. Suppression d'enregistrements : delete(objet,clé)

- ✓ On demande à X3 de supprimer un enregistrement, on lui transmet les valeurs de clés. L'ensemble des contrôles applicatifs est déclenché. C'est l'équivalent du bouton supprimer de la gestion objet
- ✓ X3 nous répond en nous donnant les messages d'erreurs éventuels, on sait si la suppression a été effectuée ou non.

4. Mise à jour d'enregistrements : modify(objet,clé,flux)

- ✓ On transmet à X3 les valeurs de clé ainsi que les champs devant être mis à jour.
- ✓ X3 effectue la mise à jour nous répond en nous donnant les messages d'erreurs éventuels. Les tables annexes sont mise à jours. C'est l'équivalent du bouton enregistrer de la gestion objet.

1.3.1 Le Service Web Objet / Méthodes

3. Exécution d'option : actionObject(objet,option,flux)

- ✓ On va demander à X3 d'effectuer une fonction liée à un menu ou un bouton ajouté à un objet. Par ce biais on peut exécuter une tâche particulière comme valoriser la commande sans la créer.
- ✓ X3 nous répond en nous donnant le contenu de chaque champ de l'écran de l'objet concerné.

1.3.2 Le Service Web Sous Programme

- Les méthodes des Sous Programmes permettent d'exécuter des traitements L4G
  - ✓ Demander un prix
  - ✓ Obtenir la photo d'un produit
  - ✓ Interroger le stock
  - ✓ Mettre à jour des réservations client …
  - ✓ … Tout sous programme X3 est éligible

- Une fois la description du sous programme enregistré dans le dictionnaire X3, celui – ci peut être déclenché dans X3 par le web service.

Le sous programme ne doit pas nécessiter l'intervention de l'utilisateur pour s'exécuter. Il doit être autonome avec les paramètres transmis lors de son appel.

1.3.2 Le Service Web Sous Programme

- Pour exécuter un sous programme il faut utiliser la méthode **runXml(code du sous programme, flux des paramètres).**

- Exemple sur une demande de tarif.On demande à X3 d'exécuter le sous programme « TARIF » en transmettant le code de l'article, le code du client.
  - ✓ X3 nous répond en nous transmettant tous les paramètres du sous programme décrits dans le dictionnaire : code de l'article, code du client et enfin le prix.
  - ✓ Il ne reste plus qu'a exploiter décoder ce prix dans l'application cliente pour l'utiliser

1.3.3 Le Service Web Liste

- **Les méthodes des Listes permettent la consultation d'informations basées sur le paramétrage des listes gauches.**
- Exemple :
  - ✓ Demander la liste des commandes d'un client
  - ✓ Demander la liste des articles d'une catégorie donnée
- Toutes les listes gauches des objets peuvent être interrogées par ce service web.
- Les critères utilisés pour construire le filtre sont cependant limités aux colonnes présentes dans le paramétrage de la liste gauche.
- La liste interrogée est la liste principale de l'objet, les autres tiroirs ne sont pas disponibles.

1.3.3 Le Service Web Liste

- Pour interroger une liste le service WEB propose la méthode **Query (objet,clé,nombre de lignes).**
  Celle-ci interroge la liste gauche d'un objet dans X3.

- **Exemple sur la liste gauche des articles :**

  - ✓ On demande à X3 de nous donner le contenu de la liste gauche de l'objet des articles. On lui transmet les critères d'interrogation de cette liste :
    - Article commençant par L : L*
    - De la catégorie « fragile » : FRL
  - ✓ X3 nous répond en nous transmettant un tableau avec toutes les colonnes paramétrées dans la liste gauche dont les lignes correspondent aux critères demandés.

1.3.4 La Méthode GetDescription

- **Une méthode permettant de vérifier la disponibilité d'un service :**
  - Cette méthode commune permet :
    - ✓ De vérifier si un objet, une liste ou un programme a été publié en mode WEB Service.
    - ✓ De recharger la dernière version de la publication dans le cache du serveur WEB
    - ✓ D'obtenir un certains nombre d'information :
      - Pour une liste : les noms et type des colonnes
      - Pour un objet : les noms et type des champs des écrans
      - Pour un sous programme : les noms et type des paramètres

En résumé …

- Si l'on désire gérer des enregistrements dans X3, en tenant compte des règles de gestion et des contrôles applicatifs.
  - L'utilisation du service web objet est tout a fait indiqué.
  - La contrainte essentielle est bien entendu que l'objet existe dans X3 et que celui-ci n'ouvre pas de fenêtre complémentaires.
  - Aucun développement n'est à priori nécessaire dans X3, seule la conformité de l'objet est a vérifier dans ce mode d'exploitation.

- Si l'on désire consulter des listes en tenant compte des restrictions liées au paramétrage des habilitations utilisateurs et des rôles
  - L'utilisation du service web liste est une des solutions possibles mais, les informations devant être consultées par l'application tierce doivent être paramétrées dans préalablement dans X3 dans une liste gauche.
  - Un nouvel objet peut cependant être créé uniquement pour sa liste gauche

En résumé …

- Si l'on désire exécuter des tâches particulières n'étant pas disponibles sous la forme d'objet dans X3
  - Le service web sous-programme est le seul permettant de répondre cette demande.
  - La contrainte principale est que le sous programme doit être autonome (pas d'interaction avec l'utilisateur)
  - Et bien entendu qu'il soit décrit dans le dictionnaire X3

Formation Web Services Adonix

2. Principe de fonctionnement

   2.1. Notion sur les solutions Sage X3

   2.2. Zoom sur le serveur WEB

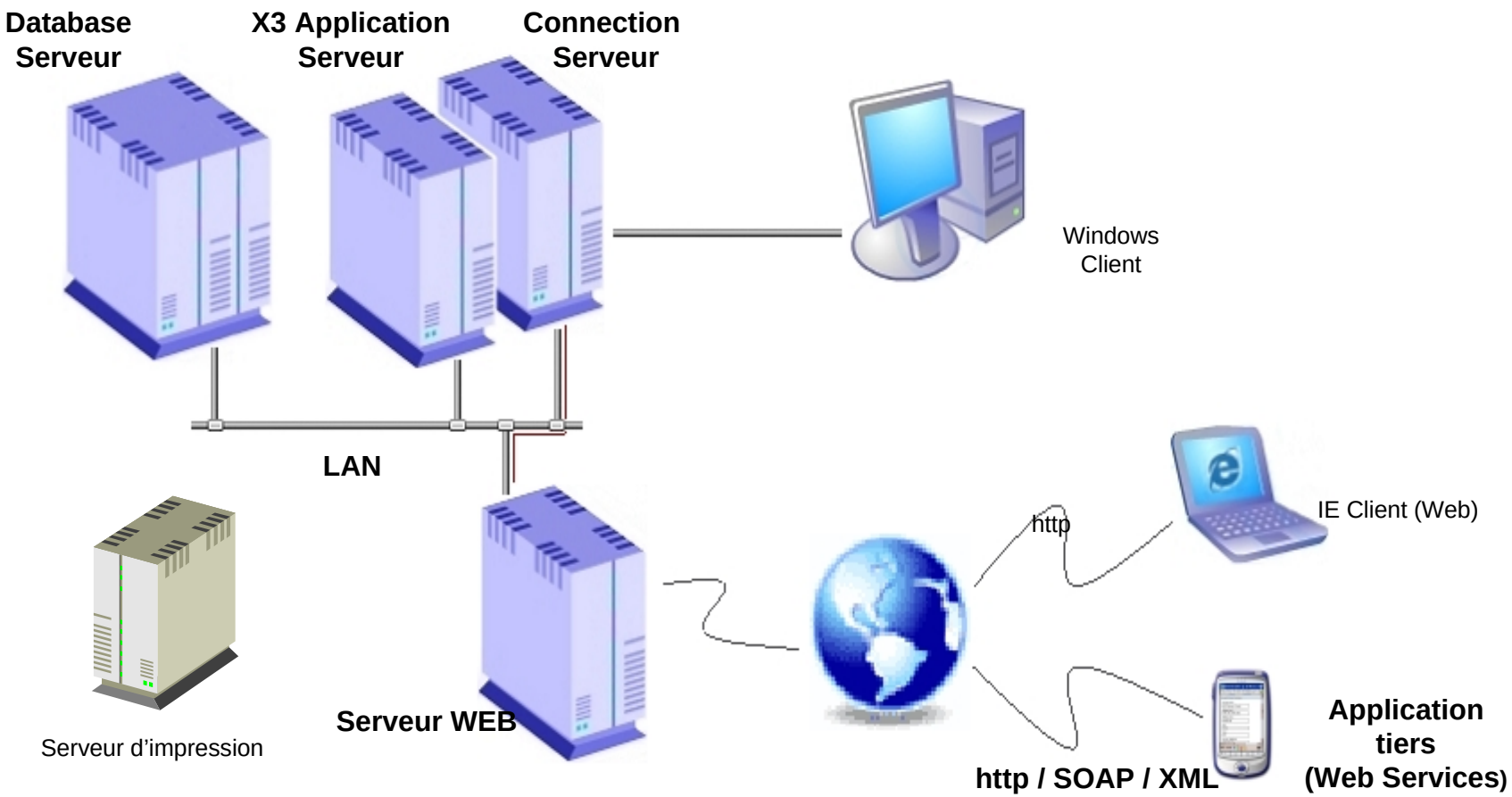
   2.3. Traitement d’une requête web / service

   2.4. Fonctionnement du pool de connexion

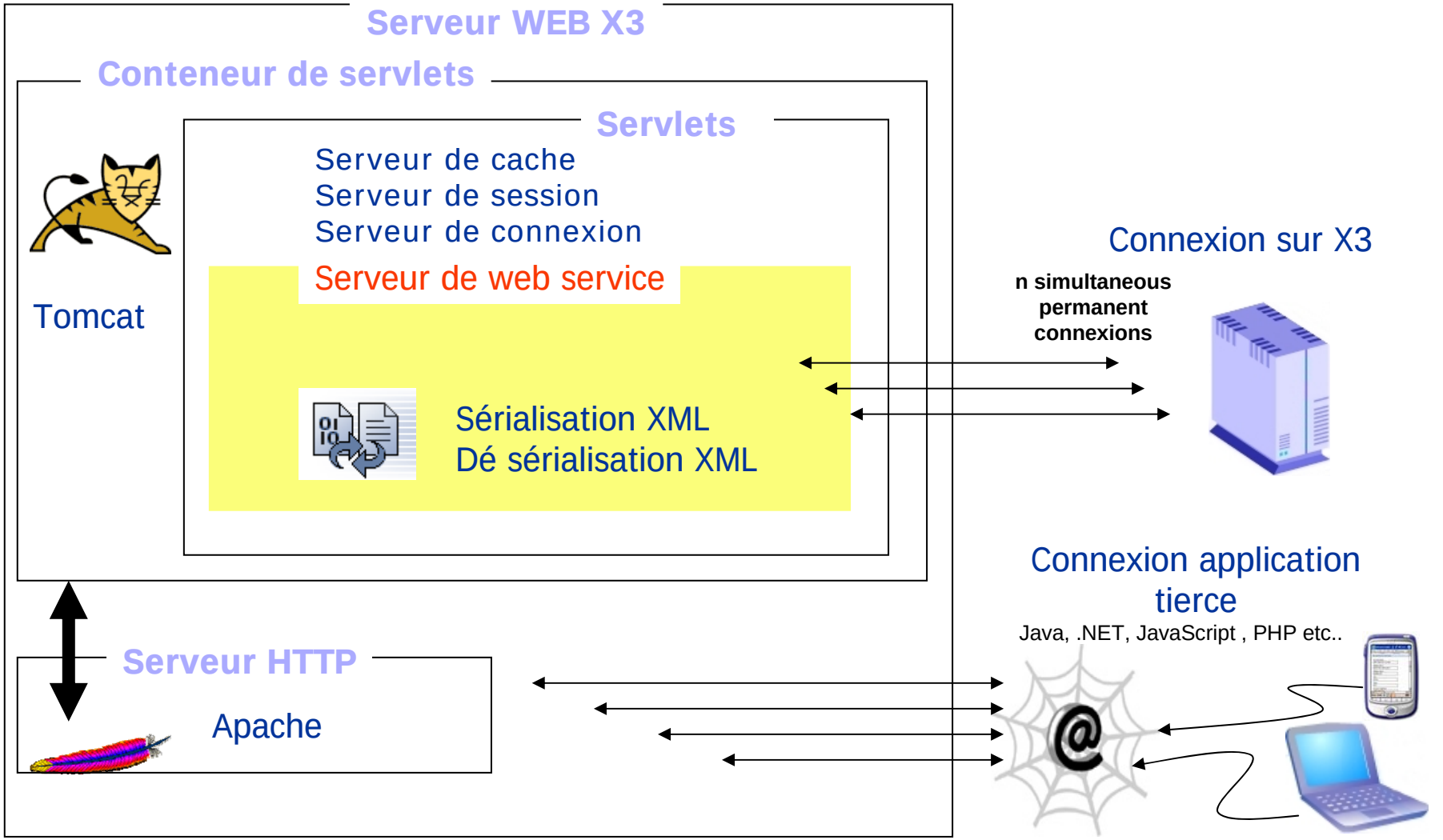
2. Notion de solution Sage X3

- Une installation Sage X3 comprend un et un seul dossier mère et n dossiers fils qui héritent des éléments définis dans les dossiers de niveau supérieur selon les règles de propagation sage X3.
  - ✓ Chaque dossier de l'arborescence possède des tables dans la base de données et un ensemble de répertoires sur disque. Il est possible de localiser les éléments sur disque sous différentes racines en utilisant le concept d'**adxvolume**.
  - ✓ Pour chaque installation le nom de dossier est discriminant, mais il est possible d'avoir sur la même machine (ou sur des machines différentes) des installations différentes comportant des dossiers de même nom que l'on souhaite publier sur le même serveur Web.
  - ✓ Il convient donc de différencier ces éléments en caractérisant chaque « installation ».
  - ✓ Une installation prend pour nom une solution Adonix et est caractérisée par un code alphanumérique défini au niveau de la console.
  - ✓ Ce code est stocké dans le fichier de configuration **solution.xml** qui se trouve au dessus du répertoire racine du dossier mère (classiquement dans le répertoire DOSSIERS sur une installation Windows)

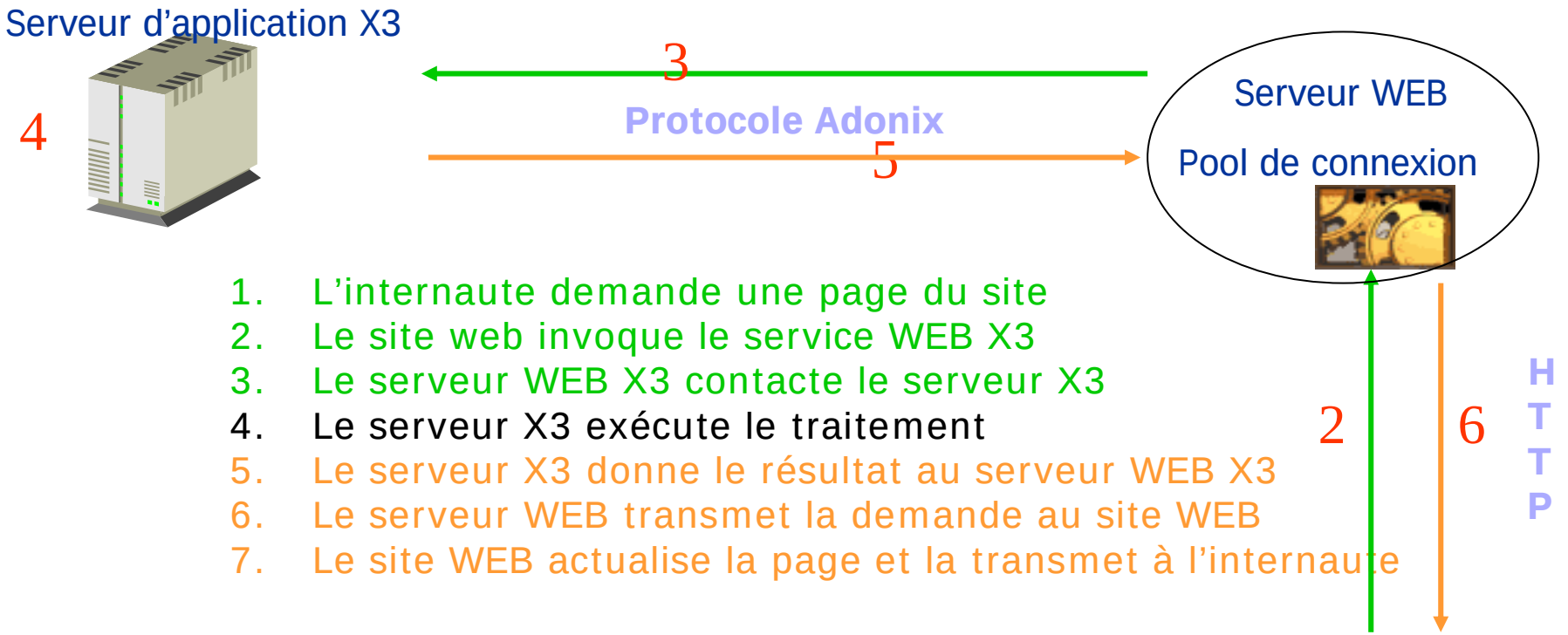
2.1 Notion de solution Sage X3



2.2 Zoom sur le serveur WEB



2.3 Traitement d'une requête Web Service



2.4 Traitement d'une requête Web Service

- **1 - L'internaute demande une page du site**
  - Il demande une page dans laquelle il y a une interaction avec X3 (création d'une commande, consultation d'un tarif …)

- **2 - Le site web invoque le service WEB X3**
  - Pour répondre à la demande de l'internaute, il faut dans le code du site WEB avoir au préalable intégré un des services WEB X3, on l'invoque en faisant varier les paramètres des méthodes qu'il contient.
  - L'invoquer reviens à construire le message SOAP et le mettre dans une requête HTTP à destination du serveur WEB X3

- **3 - Le serveur WEB X3 contacte le serveur X3**
  - Le serveur WEB X3 décode le message et le met sous une forme compréhensible par le serveur X3, il lui transmet l'information via l'une des connexions déjà établie.

- **4 - Le serveur X3 exécute le traitement**
  - Le serveur X3 exécute un programme wrapper qui se charge de garnir les champs des écrans et de déclencher le scénario objet ou d'exécuter le programme demandé

2.4 Traitement d'une requête Web Service

- **5 - Le serveur X3 donne le résultat au serveur WEB X3**
  - Pour le serveur X3 il n'y a pas de différence entre une connexion de ce type et une connexion de type client/serveur classique, il dialogue avec le serveur WEB comme il le fait avec le client graphique X3.

- **6 - Le serveur WEB transmet la demande au site WEB**
  - Le serveur WEB encode la réponse d'X3 pour la mettre sous la forme d'un message SOAP
  - Le message SOAP est encapsulé dans une requête HTTP et envoyé au site WEB

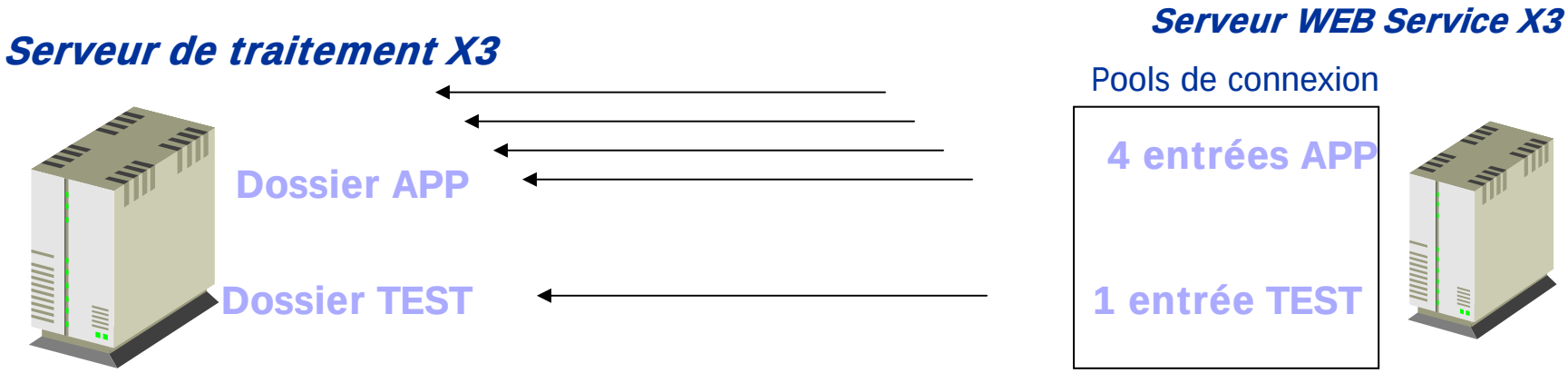
- **7 - Le site WEB actualise la page et la transmet à l'internaute**

2.5 Fonctionnement du pool de connexion

- **Le pool de connexion est le système qui permet :**
  - **D'établir les connexions sur le serveur X3**
  - **De gérer les files d'attentes des demandes**

Étape 1 - Démarrage des pools de connexion

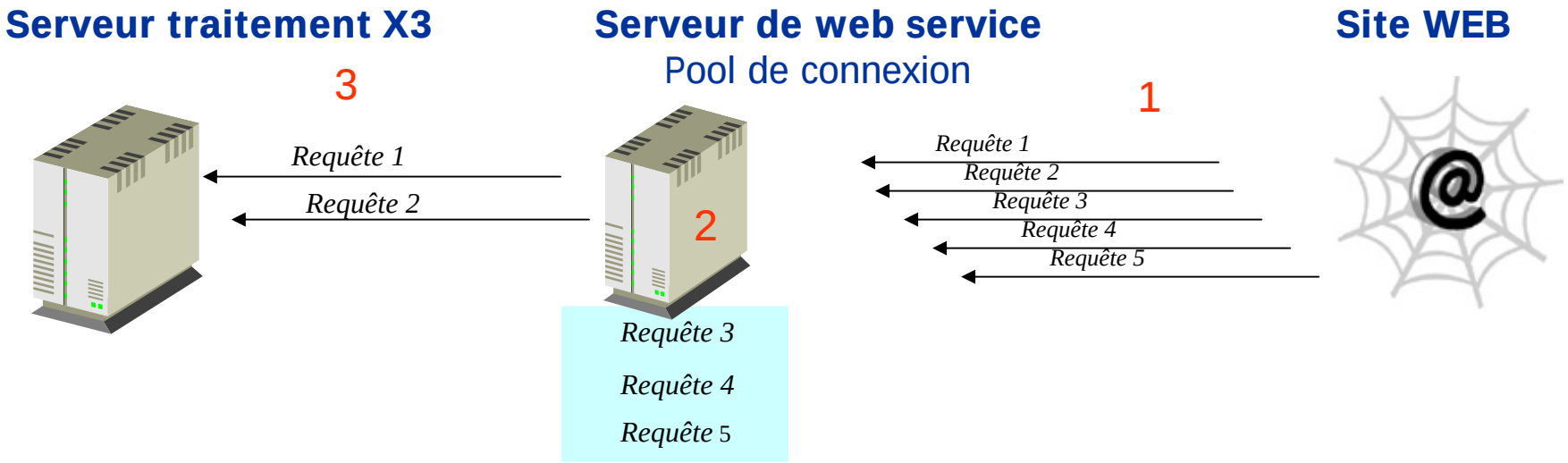
- Soit automatique lors du démarrage du serveur WEB
- Soit par une demande de démarrage explicite



Dans cet exemple les pool de connexion démarrent et lancent 4 sessions sur le serveur X3 sur le dossier APP et 1 sur le dossier TEST. Au total 5 licences de type web service sont consommées

2.5 Fonctionnement du pool de connexion

Étape 2 – Un pool est sollicité : une ou plusieurs requêtes sont arrivées



1. Plusieurs requêtes arrivent en même temps du site web
2. Le pool de connexion va distribuer les requêtes selon les connexions disponibles
   - Si aucune connexion n'est disponible le pool empile la requête
   - Les requêtes sont dépilées (FIFO) au fur et à mesure de la disponibilité des connexions
3. Le serveur WEB X3 utilise une des connexions pour traiter la demande
   - Le nombre de connexion dépend du paramétrage du serveur de web service
   - Et du nombre total de licence web service disponible sur le serveur X3

Formation Web Services Sage X3

3. Installation & configuration

   3.1. Installation du serveur

   3.2. Configuration du serveur

   3.3. La page technique du serveur de web service

3 Installation & configuration
Versions V140

Le serveur de web service est une des applications du serveur WEB X3 (servlet), il faut donc que le serveur WEB X3 soit préalablement installé. Son installation consiste à :

- Sur le serveur Web Adonix V140
  - Installation des composants freeware
    - ✓ Java Software Development Kit version 1.4.2_03
    - ✓ Apache Tomcat version 4.1.27
    - ✓ Apache HTTP Server version 2.0.48
  - Dépose de l'application Web 140
- Sur le serveur d'application X3
  - Installation Apache HTTP Server version 2.0.48
- Configuration à faire depuis la Console
  - ➢ Publication de la solution
    - ✓ Déclarer le nom de **l'Alias de la solution** à publier
    - ✓ Spécifier le chemin d'installation du serveur HTTP installé sur le serveur d'application X3
  - ➢ Configuration du serveur Web

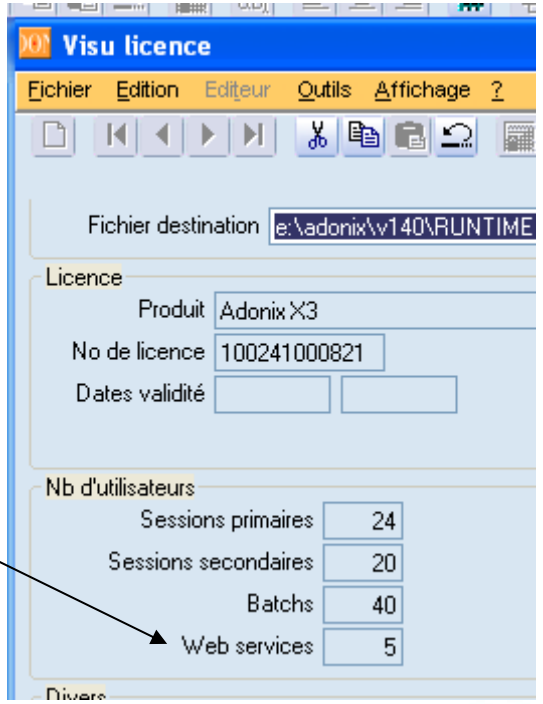
3 Installation & configuration
Version V5.0

- Sur le serveur Web Adonix V50
  - Installation des composants freeware
    - ✓ Java Software Development Kit version 1.4.2_03
    - ✓ Apache Tomcat version 5.5.20
    - ✓ Apache HTTP Server version 2.0.59
  - Dépose de l'application Web V5
- Sur le serveur d'application X3
  - Installation Apache HTTP Server version 2.0.59
- Configuration à faire depuis la Console
  - ➢ Publication de la solution
    - ✓ Déclarer le nom de **l'Alias de la solution** à publier
    - ✓ Spécifier le chemin d'installation du serveur HTTP installé sur le serveur d'application X3

3.1 Installation du serveur

- Une fois le serveur WEB installé
  - Si le dossier était déjà accessible en mode WEB il faudra juste s'assurer de sa version.
  - Il faut publier les différents dossiers, en configurant le serveur WEB
  - Cette opération est à réaliser en utilisant la console (cf. cours installation)

- Licence Web service
  - Le décompte des licences se fait séparément des sessions primaires



3.1 Installation du serveur

Version minimum pour utiliser les web services

- Applicatif X3 :Version 143 Liste de patch N°11
- Serveur WEB : 14W20
- Runtime X3 : 14R318
- Console : 14S25

Version conseillée pour utiliser les web services

- Applicatif X3 :Version 143 Liste de patch N°15
- Serveur WEB : 14W31
- Runtime X3 : 14R319
- Console : 14S35

- Dans cette version une supervision a été ajoutée pour tenir compte des architectures client qui incluent un firewall.

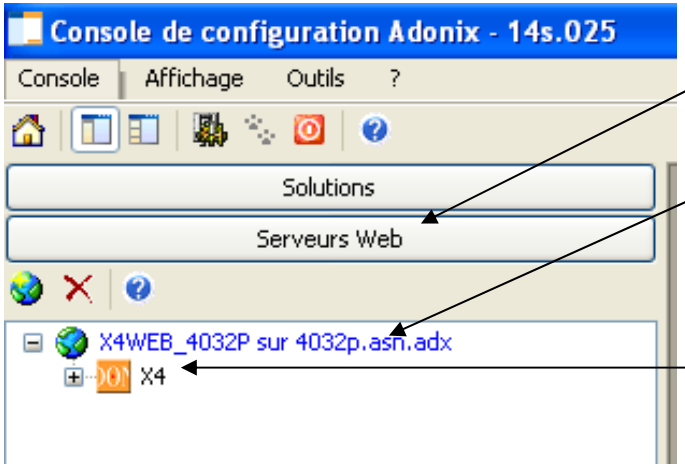
3.2 Configuration du serveur

Le serveur de web service doit être au préalable configuré pour être utilisé, cette opération est a réaliser depuis la console de configuration Sage X3

Raccourci bureau

- Le concept de déposez – cliquez
- Paramétrage facilité des composants
- Assemblage et administration centralisés et simplifiés
  - d'une architecture intégrée (un portable par exemple)
  - à une architecture multi-tiers élaborée

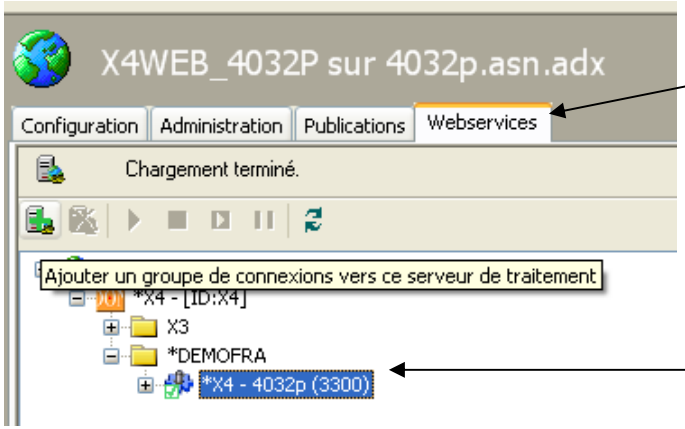
3.2 Configuration du serveur



1 Lancer la console de configuration, choisir «Serveurs WEB»

2 Choisir le serveur concerné

3 Choisir la solutions concernée

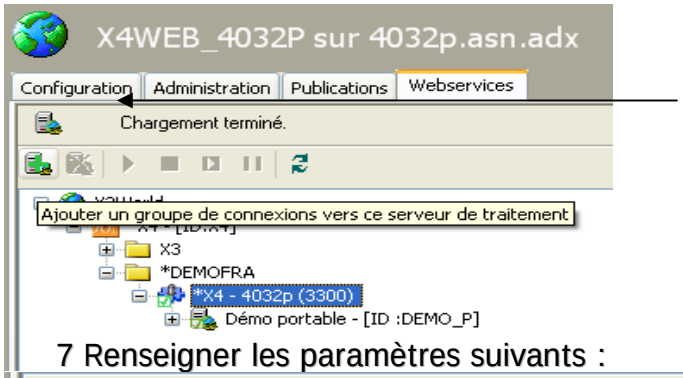


4 Cliquer sur l'onglet «Webservices» **en V140**

Ou en **V5 utiliser l'icône**

5 Choisir le dossier sur lequel les web services seront utilisés

3.2 Configuration du serveur



6 Cliquer sur « + » pour ajouter un nouveau groupe de connexion

7 Renseigner les paramètres suivants :

Nom	Valeur
alias	DEMO_P
libellé	Démo portable
solution	X4
serveur	4032P
port	3300
dossier	DEMOFRA
langue	FRA
utilisateur Adonix	ADMIN
mot de passe Adonix	
utilisateur système	Administrateur
mot de passe système	CRYPT:vdosgrRdosmTxusva
taille du groupe	5
taille initiale du groupe	2
connexion automatique	on
entrées réservées	on

Alias : Nom donné au groupe de connexion

Libellé : Intitulé du groupe de connexion

Langue : Code langue X3 utilisé pour le démarrage du groupe

Utilisateur Adonix : Code user X3 utilisé pour le démarrage du groupe

Mot de passe Adonix : Mot de passe user X3 utilisé pour le démarrage du groupe

Utilisateur système : Code de l'utilisateur réseau (compte windows)

Mot de passe système : Mot de passe du user réseau

Taille du groupe : Nombre de connexion simultanée maximum possible

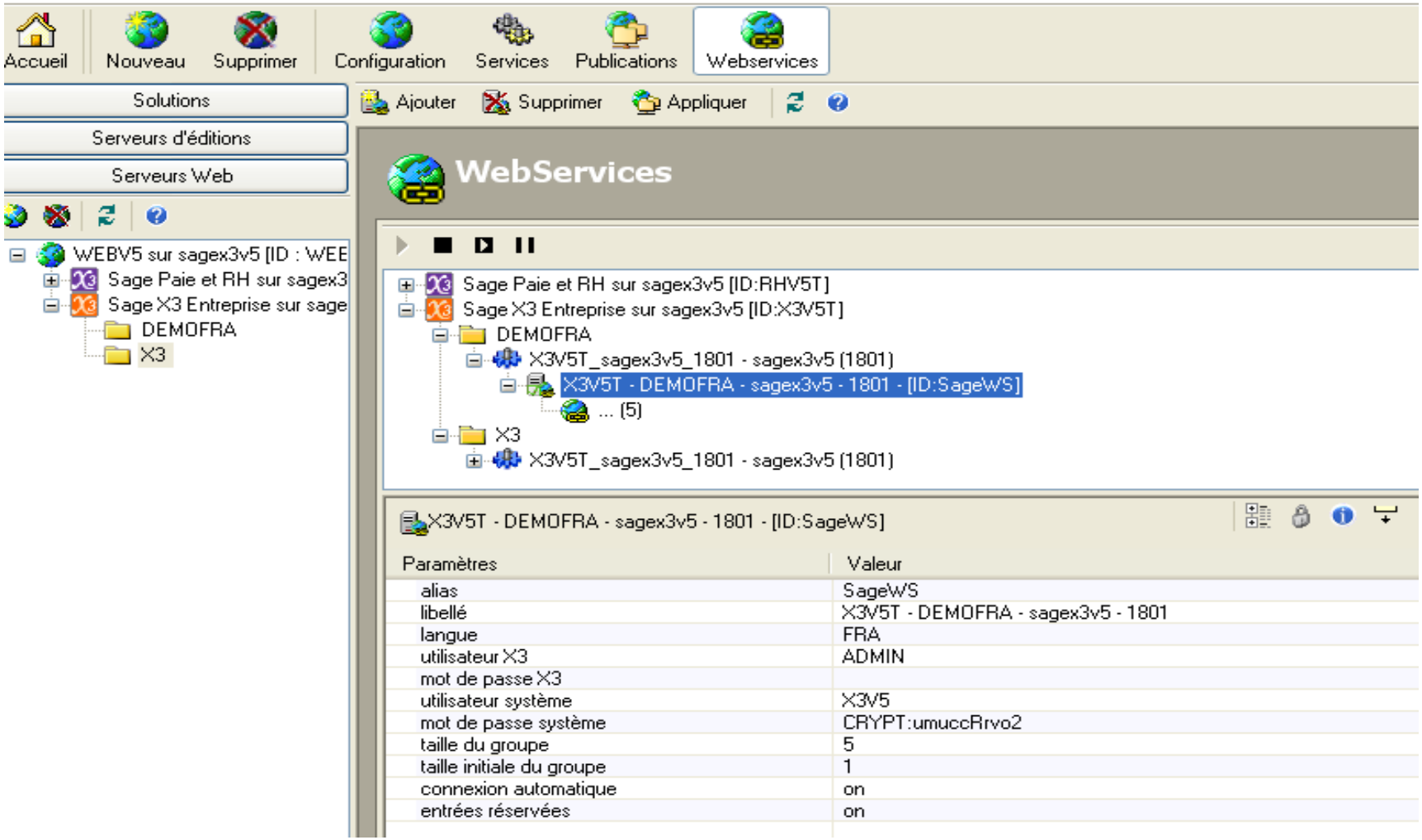
Taille initiale du groupe : Nombre de connexion ouverte dès le démarrage du groupe

Connexion automatique : Démarrage automatique du groupe de connexion

Entrée réservée : Obligation de signer chaque requête

8 Cliquer sur Enregistrer, la console configure le serveur de WEB service

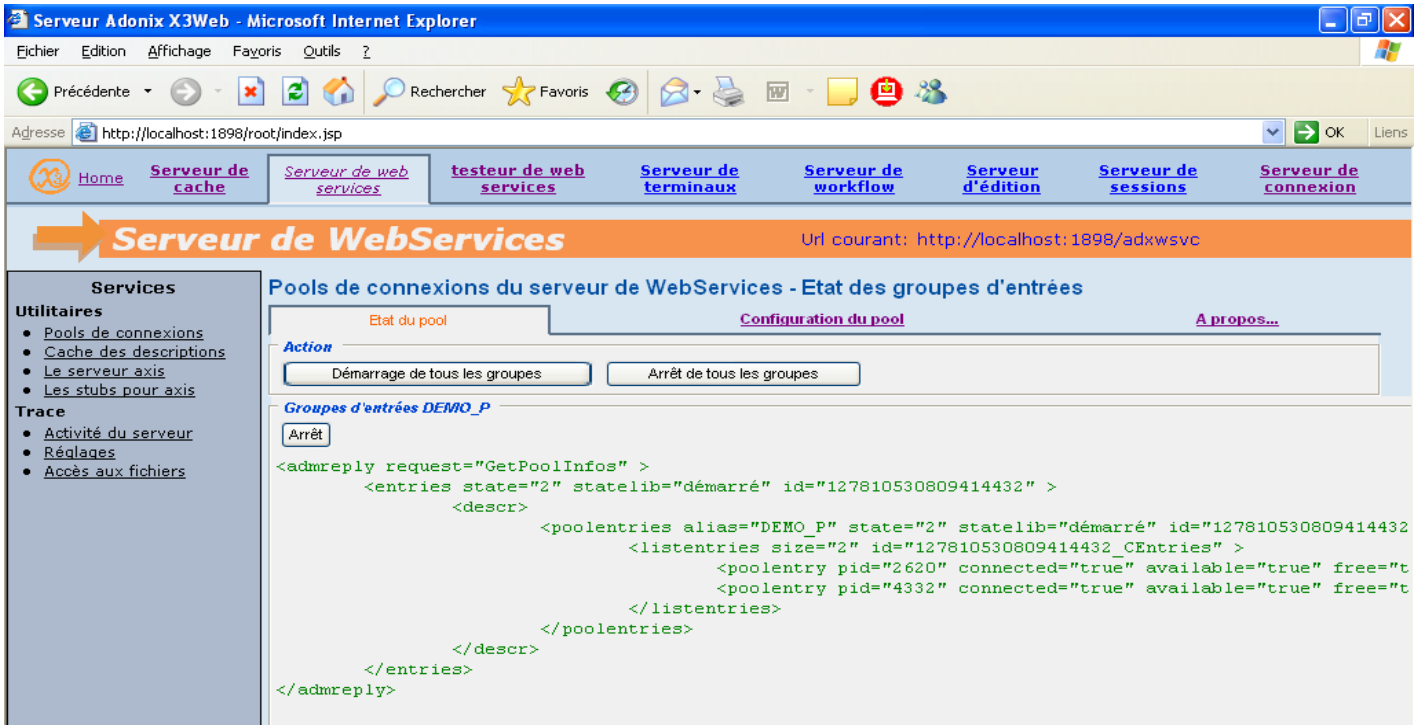
3.2 Configuration du serveur



3.3 La page technique du serveur

Pour vérifier que le serveur de web service est opérationnel, il faut :

- ouvrir la page technique avec l'url :http://localhost:1898
- et cliquer sur l'onglet Serveur de Web Service
  - Une page comme celle ci doit apparaître :

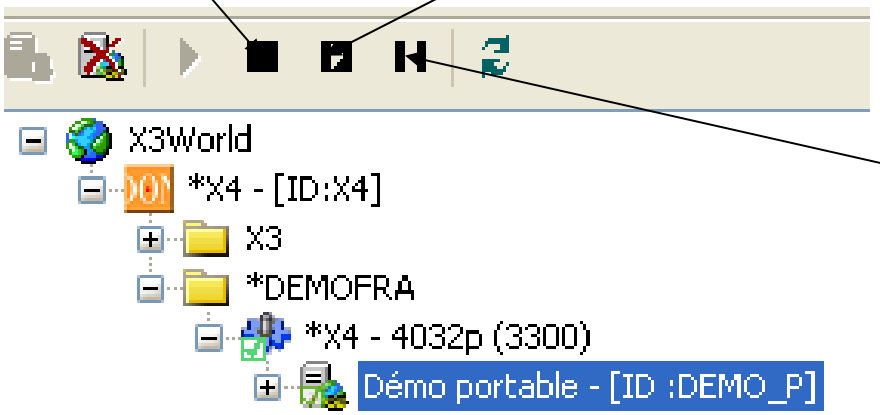


Le **démarrage** des groupes d'entrées peut se faire depuis cette page ...

3.3 La page technique du serveur

… ou depuis la console

Arret du groupe

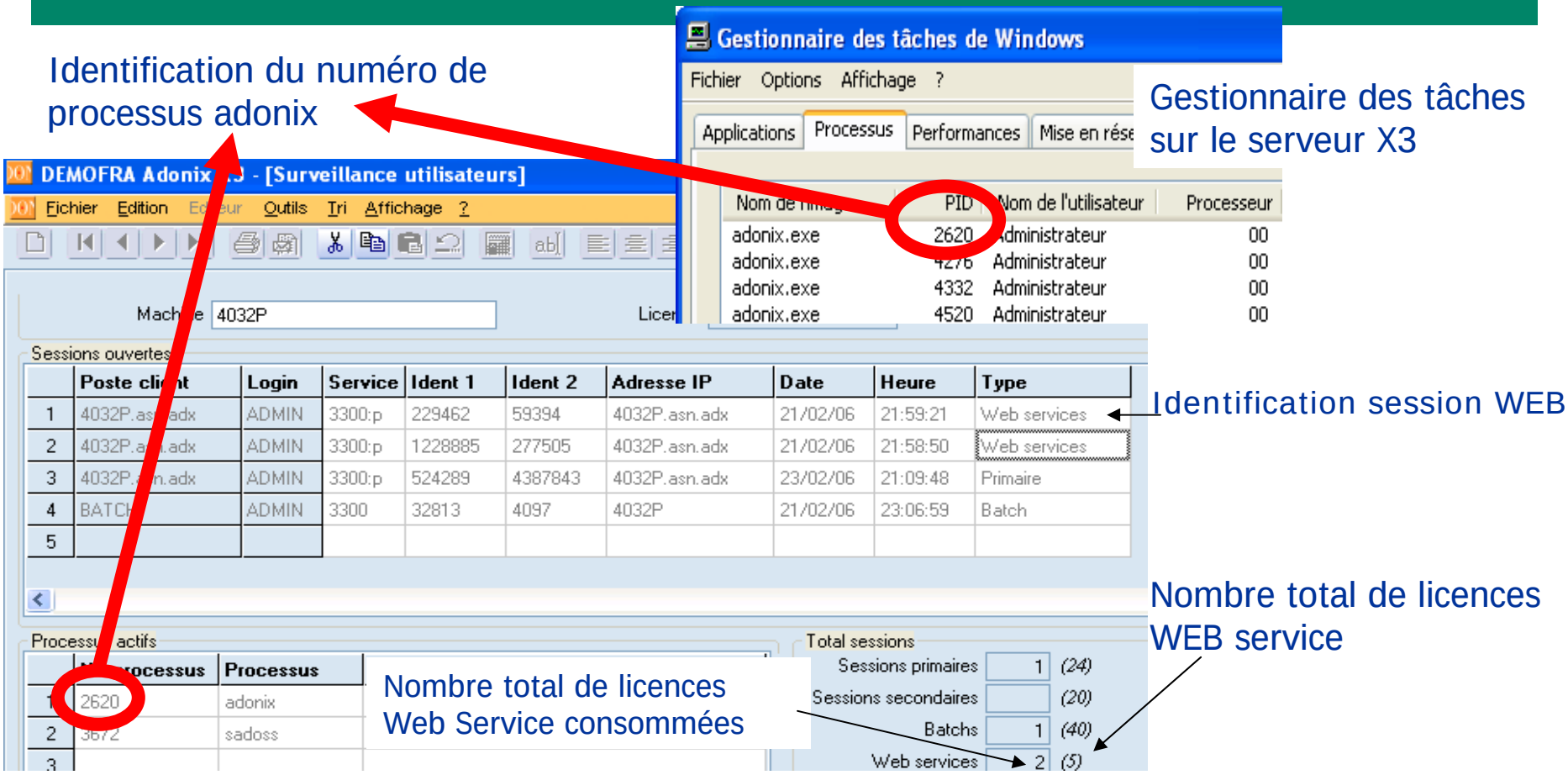


Suspension du groupe de connexions

3.3 La page technique du serveur

Identification du numéro de processus adonix

Gestionnaire des tâches sur le serveur X3



Identification session WEB

Nombre total de licences WEB service

Nombre total de licences Web Service consommées

```
<admreply request="GetPoolInfos" >
        <entries state="2" statelib="démarré" id="127810530809414432" >
                <descr>
                        <poolentries alias="DEMO_P" state="2" statelib="démarré" id="127810530809414432
                                <listentries size="2" id="127810530809414432_CEntries" >
                                        <poolentry pid="2620" connected="true" available="true" free="t
                                        <poolentry pid="4332" connected="true" available="true" free="t
```

Etat du groupe de connexion

Etat du pool de connexion

Web Services Sage X3

4.1 La modélisation dans X3

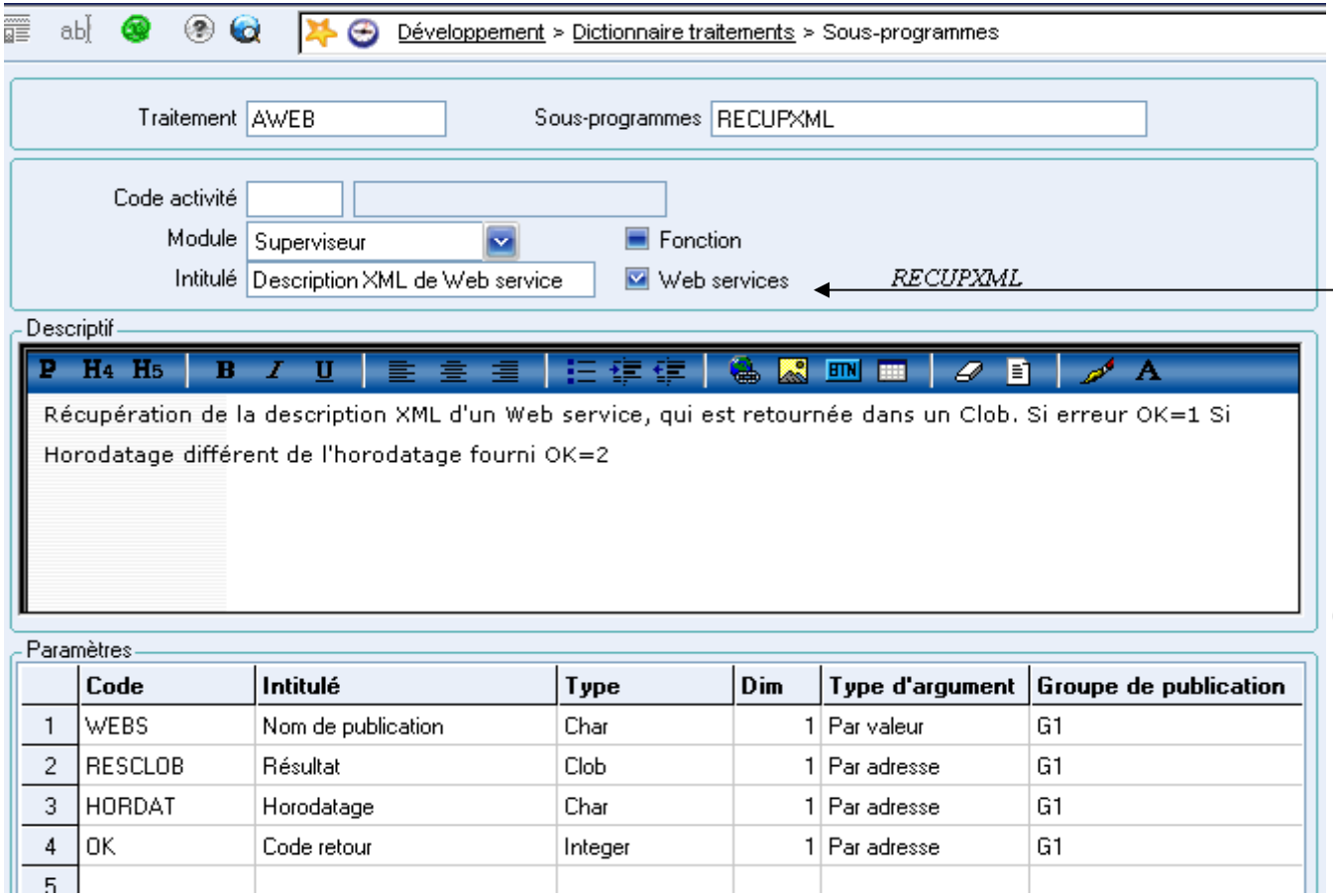
Pour utiliser des fonctions d'X3 dans le mode web services, il faut au préalable que ces fonctions soit modélisées.

Les modèles compatibles avec les web services sont :

- Les sous-programmes
- Les objets
  - Celui-ci permet l'utilisation des services web Objet et Liste
  - Ce modèle permet d'être indépendant du mode d'utilisation :
    - ✓ Client windows
    - ✓ Client Internet explorer
    - ✓ Import de données
    - ✓ Web Service

4.1.1 Modélisation des sous-programmes

Dictionnaire des traitements



	Code	Intitulé	Type	Dim	Type d'argument	Groupe de publication
1	WEBS	Nom de publication	Char	1	Par valeur	G1
2	RESCLOB	Résultat	Clob	1	Par adresse	G1
3	HORDAT	Horodatage	Char	1	Par adresse	G1
4	OK	Code retour	Integer	1	Par adresse	G1
5						

Case à cocher WEB service : Indique que le sous programme est utilisable dans ce mode

dictionnaire :

chaque

Les paramètres peuvent être passés par valeur ou par adresse
Si ils sont passés par adresse X3 peut transmettre des informations à l'application tierce

4.1.1 Modélisation des sous-programmes
Les informations gérées

- Les sous-programmes destinés à être publiés sous forme de Web services sont cochés « Web ».
- Les variables définissant l'interface du sous-programme sont automatiquement retrouvées dans le sous-programme
  - Déclarer une variable par ligne dans le sous-programme pour garantir la récupération automatique.
- Les informations successives de même cardinalité sont automatiquement rassemblées dans le même groupe, sauf si on leur associe des noms de groupe différents.
- La colonne longueur permet de dimensionner les variables de type caractère, blob et clob, définissant ainsi une taille de réceptacle pour les blob et clob. Ceci permet à l'application cliente d'effectuer un contrôle supplémentaire.
- Un bouton « publication » permet de se débrancher directement sur la fonction de publication.

4.1.1 Modélisation des sous-programmes
La gestion des messages

Des sous-programmes permettent de gérer de multiples messages pour l'exécution des sous-programmes en web services (jusqu'à 50 messages)

- Call MESSAGE(MESSAGE) from GESECRAN
  Ajoute un message d'information
- Call ADDMESSWARN(MESSAGE) from AWEB
  Ajoute un message d'avertissement
- Call ERREUR(MESSAGE) from GESECRAN
  Ajoute un message d'erreur

Les messages sont transmis dans le flux XML de réponse et sont exploitables dans l'application cliente.

4.1.1 Modélisation des sous-programmes
Les limites

- 50 variables au maximum pour un sous-programme, mais toutes les variables peuvent être dimensionnées.
- La table de description des sous-programmes est une table système (c'est la même pour tous les dossiers de la solution).
- 50 messages maximum.
- Les programmes qui sont modélisés ne doivent pas ouvrir de fenêtre pour faire saisir des informations à l'utilisateur. Il doit être autonome avec les paramètres fournis.
- Toutes les instructions liées aux affichages sont à proscrire, en revanche toutes les instructions de lecture et de mise à jour de la base de données sont bien entendu disponibles.

4.1.2 Modélisation des objets
Les différents types de connexions

- A la connexion dans toute application développée en technologie Adonix on distingue les types de connexion suivants :
  - Session primaire — adxtyp=1 (C/S) ou 9 (Web)
  - Session secondaire — adxtyp=2 (C/S) ou 10 (Web)
  - Batch — adxtyp=3
  - Terminal VT — adxtyp=5
  - Web service — adxtyp=12

- En fonction des droits portés par la clé Adonix, la connexion est autorisée ou non. En particulier le serveur de Web services ne pourra pas démarrer si la clé n'autorise pas au moins 1 connexion.

- En mode Web service la variable globale numérique GWEBSERV est égale à 2

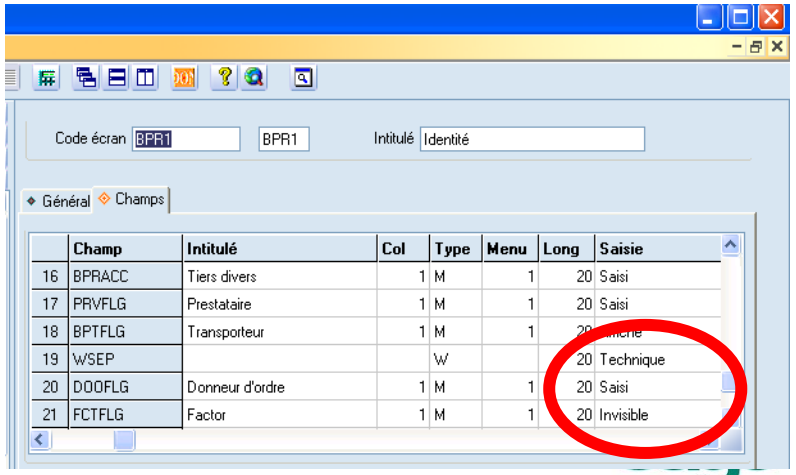
4.1.2 Modélisation des objets

- Les objets X3 sont bien entendu eux aussi stockés dans des dictionnaires dans lesquels on retrouve :
  - La description de la liste gauche
  - La description de chacun des écrans qui le compose
  - La description des options particulières
  - La table principale à mettre à jour
  - Les traitements L4G qui sont liés à cet objet
- Le dictionnaire n'est, à priori, pas à compléter. Si l'objet est disponible en client/serveur il l'est aussi en mode web service. Cependant de légères adaptations peuvent être nécessaire pour accéder aux écrans de saisie complémentaires (adresse dans la commande par exemple).
- L'objet est le modèle le plus couramment utilisé dans X3, les fonctions qui n'ont pas été développées avec ce modèle sont en général encapsulable dans un sous programme à réaliser en spécifique.

4.1.2 Modélisation des objets
Mode d'affichage des champs

- Une zone peut être :
  - Saisie
    - ✓ Que ce soit en web service ou en C/S la zone sera transmise au client
  - Affichée
    - ✓ Que ce soit en web service ou en C/S la zone sera transmise au client
  - Invisible
    - ✓ En C/S la zone n'est jamais transmise au client, en web service cela dépend du mode de publication (zone invisible oui/non)
  - Technique
    - ✓ Que ce soit en web service ou en C/S la zone ne sera jamais transmise au client

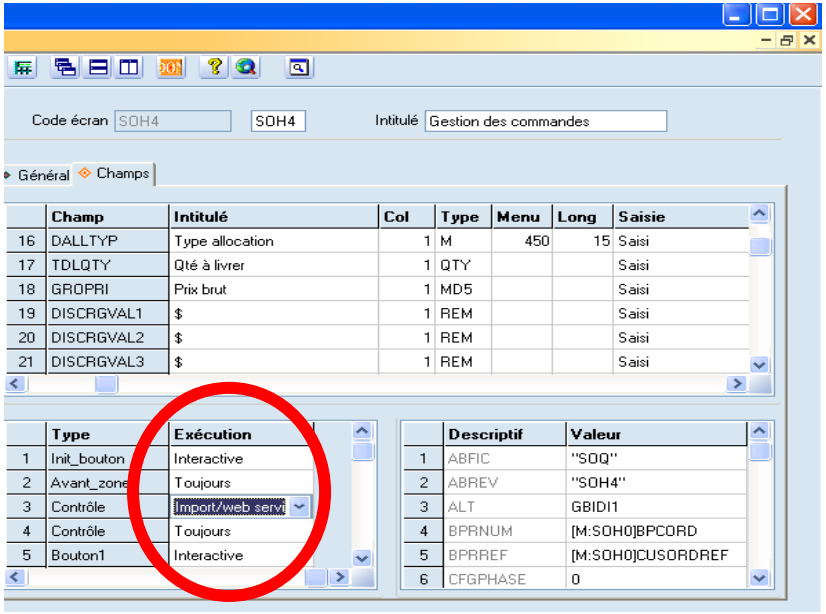
Les zones non transmises au client sont bien entendus disponibles dans la classe [M]
Cela évite de grossir inutilement les flux client/serveur



4.1.2 Modélisation des objets
Mode d'exécution des actions

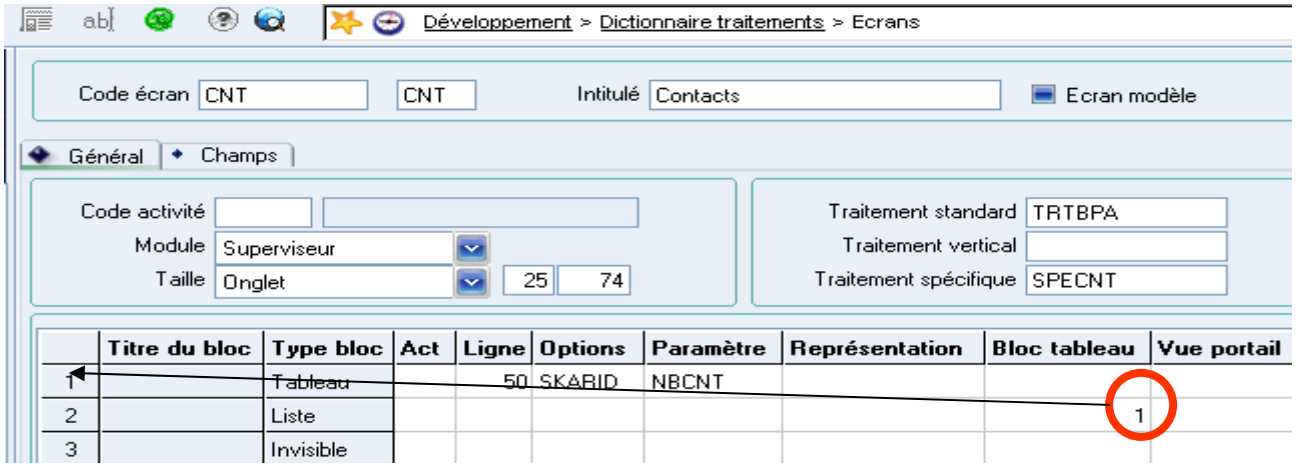
- Une action peut être exécutée
  - Interactive
    - ✓ Action exécutée uniquement en C/S
  - Toujours
    - ✓ Que ce soit en web service ou en C/S l'action est exécutée
  - Import / Web service
    - ✓ Action exécutée uniquement en web service et en import

- Ne sont jamais exécutées :
  - Les actions avant bouton
  - Les actions Bouton N
  - Les actions de sélection
  - Clic sur une icône



4.1.2 Modélisation des objets
Les écrans liste / détail

- Modélisés dans le dictionnaire (exemple contact dans la fiche client)
- Donnent lieu à une génération particulière, mais doivent respecter :
  - une normalisation des noms de champs (nom champ détail=«constante»+nom champ invisible dans bloc tableau
    - ✓ **exemple KCNTSRV=«K»+CNTSRV)**
  - Le dernier champ de type numérique (C) et invisible du bloc détail est considéré comme la variable de contrôle du numéro de ligne dans le tableau et est initialisée en tant que telle par le programme wrapper lors de l'appel de chaque ligne, exemple : KNOLIG dans CNTBPC.
  - Le bloc liste doit faire référence au bloc tableau



4.1.2 Modélisation des objets
Les écrans liste / détail

	Champ	Bloc	Pos	Intitulé
4	CNTFNA	1	4	Prénom
5	CNTCRY	1	5	Pays
6	CNTFLG	1	6	Par défaut
7	CNTTTL	1	7	Civilité
8	CNTFNC	1	8	Fonction
9	CNTSRV	1	9	Service
10	CNTADD	1	10	Adresse
11	CNTTEL	1	11	Téléphone
12	CNTFAX	1	12	Fax

	Champ	Bloc	Pos	Intitulé
20	KCCNCRM	2	1	Code
21	KCNTTTL	2	2	Civilité
22	KCNTLNA	2	3	Nom
23	KCNTFNA	2	4	Prénom
24	KCNTFNC	2	5	Fonction
25	KCNTSRV	2	6	Service
26	KCNTMSS	2	7	Mission
27	KCNTBIR	2	8	Date naissance
28	KCNTADD	2	9	Adresse

36	KNOLIG	2	20	No	1	C	2	Invisible

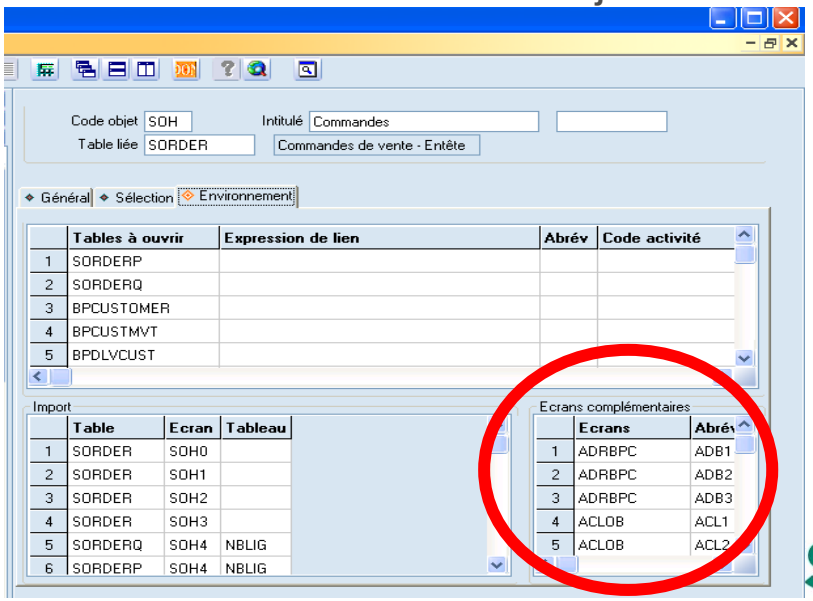
- Seuls les champs du tableau sont exportés en XML. Les champs invisibles du bloc tableau sont automatiquement ajoutés.
- Le programme de contrôle déverse les lignes du tableau dans le bloc détail ligne par ligne et effectue les contrôles sur champ définis dans l'écran
- Attention :
  - Les actions du tableau ne sont pas exécutées
  - Il faut en cas d'action MODIFY, et de modification dans ce tableau, charger la zone précisant le nombre de lignes du tableau (exemple NBCNT dans les contact), faute de quoi c'est le plus grand numéro de ligne indiqué qui est pris comme dimensionnement du tableau

4.1.2 Modélisation des objets
Les écrans complémentaires

- Dans le dictionnaire Objet, il est possible de définir de 0 à 8 écrans complémentaires avec une abréviation associée.
- Ces écrans vont être
  - Pris en compte dans la structure XML produite lors de la génération du Web service, comme des écrans de la fenêtre
  - Ouverts par le programme automatique associé à la fenêtre
  - Déclarés comme des onglets de la fenêtre lors de l'ouverture de la fenêtre dans l'instruction Inpbox
  - Chargés dans les variables du flux XML de réponse du Web service
  - Chargés via une instruction Saizo à partir des informations envoyées par le Web service lors d'une action de CREATE ou MODIFY
- L'applicatif devra assurer dans le contexte Web service (GWEBSERV=2)
  - Le chargement des masques complémentaires (étiquette lien)
  - La mise à jour des tables à partir des masques lors de l'écriture

4.1.2 Modélisation des objets
Les écrans complémentaires

- Une évolution a été faite pour illustrer le principe de fonctionnement des écrans complémentaires. Elle concerne l'objet SOH.
- Cette évolution permet de récupérer en lecture/écriture :
  - Les adresses de commande, livraison, facturation
  - Les textes entête et pied de la commande
- Pour la mettre en œuvre il faut
  - Récupérer le source SPESOH livré à titre d'exemple et soit le mettre en ligne, soit inclure son contenu dans le SPESOH existant
  - Définir 5 masques complémentaires comme suit dans l'objet SOH
    - ✓ ADRBPC ADB1
    - ✓ ADRBPC ADB2
    - ✓ ADRBPC ADB3
    - ✓ ACLOB ACL1
    - ✓ ACLOB ACL2

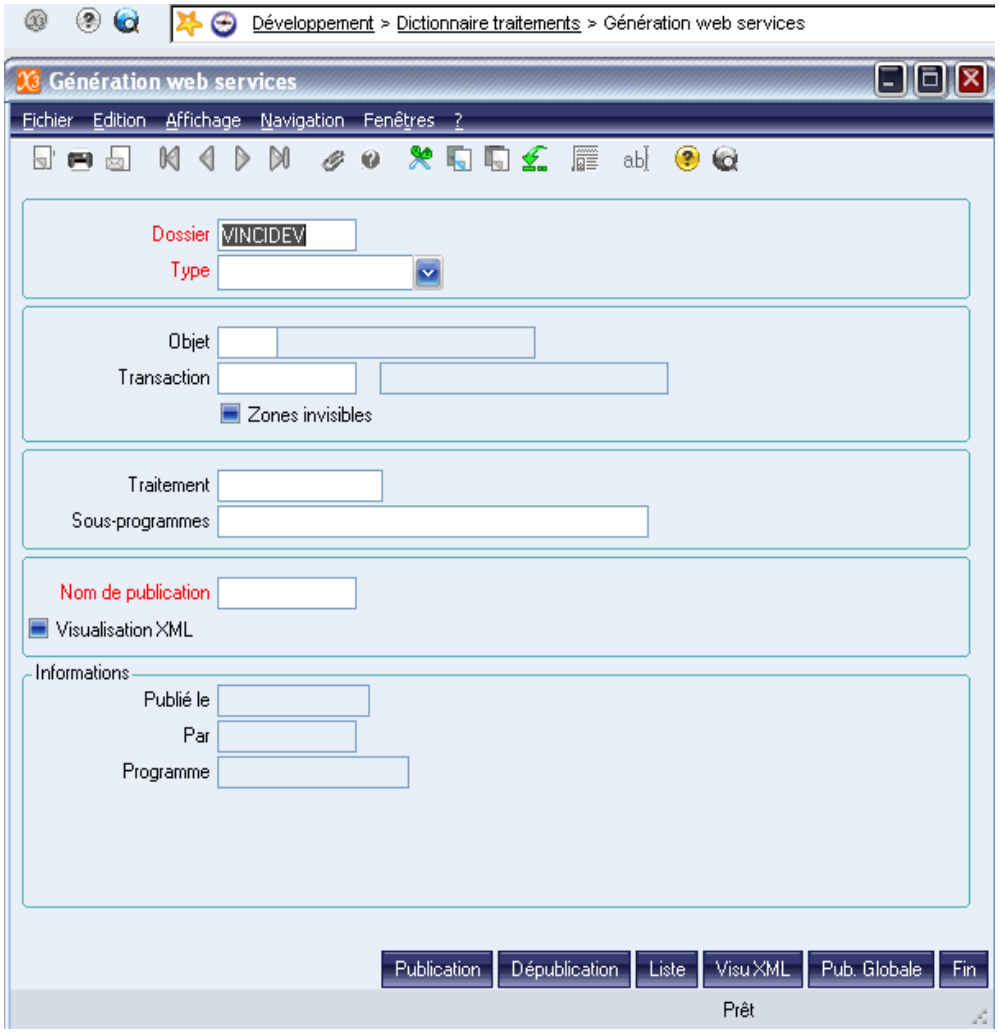


4.2 Publication des WEB services

Programme Wrapper

- Une fois l'objet ou le sous programme décrit dans le dictionnaire, il faut le publier.
- La publication permet de générer un flux XML décrivant l'objet ou le sous programme.
- Ce flux sera utilisé ensuite par le serveur web pour décoder les requêtes émanant de l'application tierce.
- Les web service X3 étant génériques, les éléments variables tels que les noms des champs pour un objet ou les paramètres pour un sous programme, sont passés sous la forme d'un flux XML, la structure de celui-ci est déterminé par la description donnée dans le dictionnaire.
- La publication génère aussi un **programme wrapper** en L4G adonix
  - Il sert d'interface entre le développement X3 et le serveur de Web services et gère:
    - ✓ Les paramètres d'entête
    - ✓ Les messages
    - ✓ Les flux montant et descendant de données
- Les programmes wrapper sont appelés par le serveur de Web services dont le rôle est de sérialiser / désérialiser le flux soap de / vers les paramètres attendus.

4.2.1 Le générateur de WEB services
Fonction de publication



4.2.1 Le générateur de WEB services
Fonction de publication

- Cette fonction permet de publier des objets avec pour chacun le choix
  - de la transaction (si cet objet a des variantes)
  - de la publication ou non des champs invisibles (les champs de type technique ne sont jamais publiés)
- Il est également possible :
  - de choisir le nom de publication (nom public de l'objet ou du sous programme)
  - de lister les éléments publiés
  - de dé-publier des éléments précédemment générés
  - de générer tous les Web services d'un dossier (bouton publication globale)
- Il est à noter que lorsqu'un nom de publication a été donné à un élément (objet ou sous-programme), ce nom est systématiquement re-proposé lors des générations suivantes. Il n'est pas possible de publier 2 fois le même couple objet-variante avec deux noms différents, ni de publier un couple objet-variante avec et sans les champs invisibles.

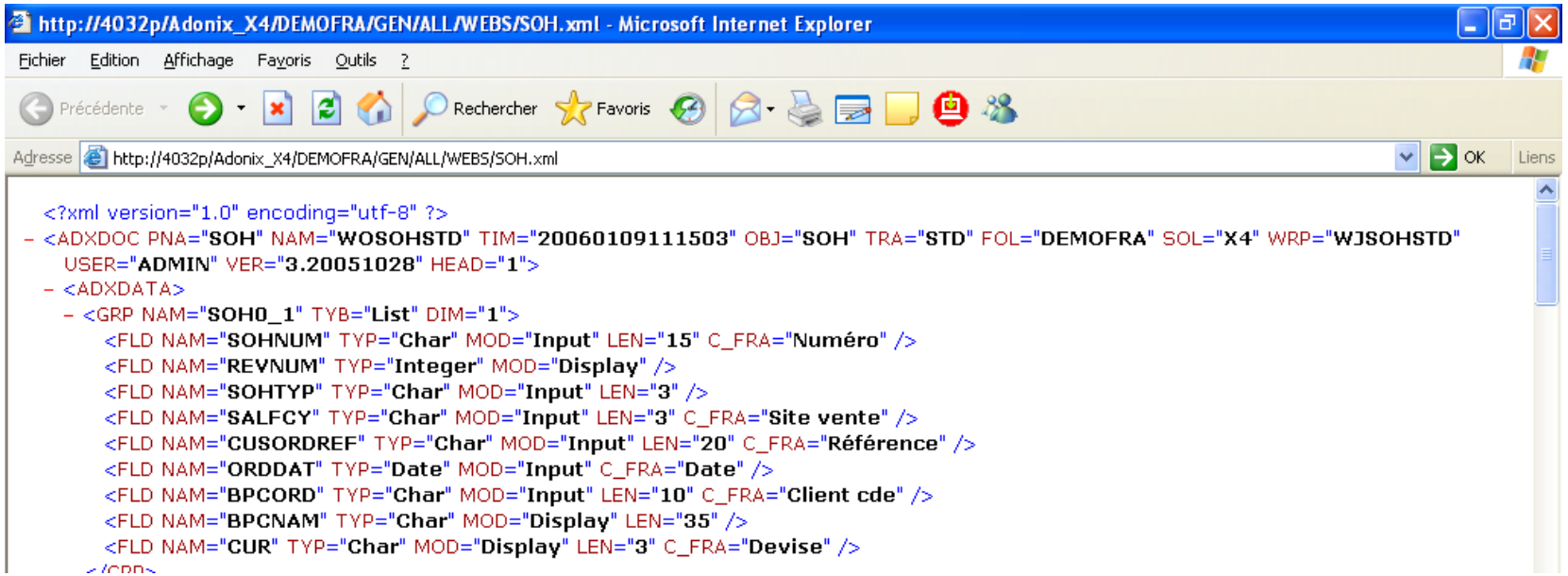
4.2.1 Le générateur de WEB services
Fonction de publication

- Le programme de publication crée :
  - une structure XML de nom «Nom de publication».xml qui est stockée dans le répertoire X3_PUB/ «Nom du dossier»/GEN/ALL/WEBS
  - un programme wrapper en L4G

- Avec chaque programme wrapper un programme de test automatique est crée qui permet d'appeler un sous-programme publié et d'appeler un objet par une des quatre méthodes proposées (lecture, création, modification, suppression) et par la méthode liste.

- Les éléments générés sont horodatés et un contrôle est effectué par le serveur de Web services avant tout appel à un programme wrapper pour s'assurer que la version publiée et utilisée est bien la plus à jour.

4.2.2 Description XML générée



Le fichier XML produit est écrit en codage UTF8, ce fichier XML peut être demandé par l'application cliente (méthode GetDescription) qui peut l'exploiter, pour générer des écrans par exemple.
L'entête du document est décrite dans le nœud ADXDOC.
Ses propriétés sont les suivantes :

4.2.2 Description XML générée Nœud <ADXDOC>

Entête du fichier XML OBJET :

<ADXDOC PNA="**OBJSOH**"	Nom de publication de l'objet
NAM="**WOSOHSTD**"	Nom de la fenêtre associée à l'élément
TIM="**20060109111503**"	Horodatage de génération
OBJ="**SOH**"	Nom de l'objet généré
TRA="**STD**"	Nom de la transaction choisie
FOL="**DEMOFRA**"	Nom du dossier
SOL="**SOLX3**"	Nom de la solution Adonix
WRP="**WJSOHSTD**"	Nom du programme wrapper
USER="**ADMIN**"	Utilisateur X3 ayant déclenché la génération
VER="**3.20051028**"	Version du générateur
HEAD="**1**">	Version du protocole d'encapsulation

4.2.2 Description XML générée Nœud <ADXDOC>

Entête du fichier XML Sous programme :

<ADXDOC PNA="**STOCKSITE**“	Nom de publication du sous programme
TIM="**20060208121856**„	Horodatage de génération
PRG="**ZSTOCK**"	Nom du programme publié
SPG="**STOCK**"	Nom du sous programme publié
FOL="**DEMOFRA**"	Nom du dossier
SOL="**SOLX3**"	Nom de la solution Adonix
USER="**ADMIN**"	Utilisateur X3 ayant déclenché la génération
VER="**3.20051028**"	Version du générateur
HEAD="**1**"	Version du protocole d’encapsulation
WRP="**WRSTOCKSITE**"	Nom du programme wrapper
C_FRA="**Recherche de Stock**">	Désignation du sous programme

4.2.2 Description XML générée Nœud <GRP>

Le nœud GRP décrit soit un bloc écran soit un groupe de paramètres d'un sous programme :

Exemple bloc de type liste :

```
<GRP NAM="SOH0_1"      Code écran et numéro de bloc
TYB="List"             Type de bloc (valeurs possibles : Table, List, Pict, Text, Hidden)
DIM="1">               Nombre de lignes maximum Toujours 1 dans les bloc de type liste
```

Exemple bloc tableau :

```
 <GRP NAM="SOH4_1"
TYB="Table"                 Bloc de type tableau
DIM="200"                   Nombre de lignes maximum
IDTAB="SOH4~NBLIG"          Code écran + variable de bas de tableau
OPT="IAD">                  Option du bloc écran : Insertion, Annulation, Suppression
```

Exemple groupe de paramètres pour un sous programme :

```
<GRP
NAM="G1"          nom du groupe dans le dictionnaire des sous programmes
DIM="1">          Nombre d'occurences des paramètres contenus dans ce groupe
```


4.2.2 Description XML générée Nœud <FLD>

Le nœud FLD décrit soit le champ d'un écran soit un paramètre d'un sous programme :

Description d'un champ d'un écran :

<FLD NAM="**ORDSTA**"	Nom du champ
TYP="**Integer**"	Type du champ (integer, char, blob, clob,date, decimal)
MOD="**Display**"	Mode d'affichage du champ (input, display, hidden)
LEN="**3**"	Longueur,si le champ est un menu local l'attribut LEN n'est pas présent
DIM="**2**"	Nombre d'occurrences du champ,si=1 l'attribut DIM n'est pas présent
MEN="**415**"	Numéro du menu local
C_FRA="**Devise**" />	Intitulé du champ le nom de l'attribut est C_+code langue

Description d'un paramètre :

<FLD NAM="**STOCK**"	Nom du paramètre
TYP="**Decimal**"	Type du paramètre (integer, char, blob, clob,date, decimal)
PAR="**Adr**"	Mode de transmission du paramètre (Adr:par adresse, Value:par valeur)
MOD="**Input**"	Toujours input
C_FRA="**Stock ar"ticle** />	Désignation du paramètre

4.2.2 Description XML générée Nœud <ADXKEY>

Le nœud ADXKEY décrit une liste gauche

```
<ADXKEY> Liste gauche
 <GRP NAM="LEFTLIST"                 Attribut fixe
          DIM="10000">               Attribut fixe
  <FLD NAM="SOHNUM"                  Code du champ de la liste gauche
          MOD="Display"              Attribut fixe
          TYP="Char"                 Attribut fixe
          LEN="30"                   Attribut fixe
     <C_FRA="No commande"/>          Intitulé de la colonne
 </GRP>
</ADXKEY>
```


4.2.2 Description XML générée Nœuds <ADXMEN>

Le nœud ADXMEN décrit les menus locaux

```
<ADXMEN>
<MNU NO="1">
  <VAL IND="1" C_FRA="Non" />
  <VAL IND="2" C_FRA="Oui" />
  </MNU>
<ADXMEN>
```

Liste des menus locaux
Numéro du menu local
Numéro et désignation de l'item du menu local

4.2.2 Description XML générée Nœuds <ADXSER>

Le nœud ADXSER décrit les actions possibles sur un objet

```
<ADXSER>

  <MET ID="READ" C_FRA="Lire" />

  <MET ID="CREATE" C_FRA="Créer" />
  <MET ID="MODIFY" C_FRA="Modifier" />
  <MET ID="DELETE" C_FRA="Supprimer" />
  <MET ID="LIST" C_FRA="Liste" />
  <MET ID="V" C_FRA="Valorisation" />

</ADXSER>
```

Liste des boutons et options disponibles

L'attribut ID est fixe les intitulés sont par langue

Option code « V » du dictionnaire des fenêtres

4.2.2 Description XML générée Nœuds <ADXREAD>

Le nœud ADXREAD donne le champ clé

```
<ADXREAD TAB="SORDER">          Nom de la table X3
  <FLD NAM="SOHNUM"             Code du champ clé
       TYP="Char"               Type du champ clé
       LEN="15"                 Longueur du champ clé
       C_FRA="No commande" />   Désignation du champ clé
</ADXREAD>
```


4.2.3 Règles de génération

- Une fonction déjà publiée doit garder le nom initial de publication (si on désire en changer, il faut la dépublier et relancer la publication)
- Les éléments publiés sont nativement multi-langue et la structure XML est stockée dans le répertoire WEBS du dossier sous X3_PUB/GEN/ALL
- Tous les éléments générés sont automatiquement horodatés, mémorisent la version du générateur utilisé et le nom de l'utilisateur X3 qui a lancé la génération (GUSER)
- La description d'un objet ou d'un sous programme indique que celui-ci peux être utilisé par une une application tierce, en revanche rien n'oblige à utiliser ce fichier dans l'application cliente.

4.2.3 Règles de génération

- Si la description XML du Web service appelée n'est pas à jour, elle sera automatiquement rafraîchie par le serveur de Web services qui récupérera la nouvelle description XML et mettra à jour ses structures.

- Pour forcer l'utilisation de la nouvelle version du programme wrapper une instruction de flush mémoire adxmpr=adxmpr est provoquée sur les différentes entrées du pool.

- Le serveur de Web services assure la sérialisation / désérialisation des paramètres et la mise à jour des structures publiées

- Pour qu'un web service bénéficie de la modification d'un programme L4G alors que celui-ci a déjà été exécuté, il faut republier l'objet.

- L'utilisateur X3 web service ne doit pas avoir le serveur batch en démarrage automatique à la connexion

4.3 Gestion de la cohérence

- **Lors de la lecture d'une fiche objet, si les champs UPDDAT, UPDTIM et UPDUSR sont définis dans la table principale de l'objet ces informations sont retournées dans le flux XML de réponse dans le groupe ADXTEC**
  - WW_MODSTAMP Horodatage de la dernière mise à jour
  - WW_MODUSER Code utilisateur X3 du dernier modifieur

- Lors de l'envoi d'un flux de données pour modification ou suppression :
  - Si et uniquement si la zone WW_MODSTAMP est valorisée (avec le contenu obtenu lors de la lecture), le programme va contrôler si l'enregistrement a été modifié depuis ce timestamp.
  - Dans ce cas un message d'erreur sera envoyée donnant le nouveau timestamp, et les modifications seront refusées avec le message : « Enregistrement modifié depuis la dernière lecture » + timestamp de la modification + utilisateur ayant modifié (si disponible)

4.4 La mise au point

- Une fois la modélisation terminée il faut tester pour vérifier si les objets et les sous programmes publiés sont bien compatibles avec le mode web service.

- Des tests sont nécessaires, en fonction des paramétrage propre à chaque dossier, les objets peuvent réagir de manière différentes (ouverture de fenêtres complémentaires, questions posées à l'utilisateur …) .

- Les objets complètement spécifiques sont eux aussi à tester.

- Des règles de développement sont à respecter.

4.4.1 Les règles de développement

Les principales contraintes sont liées à des problèmes d'affichage :

- L'instruction infbox ne bloque pas le web service mais elle ne ne permet pas de renvoyer un message à l'application cliente

- L'utilisation de la syntaxe #@ pour accéder aux postes client n'est pas compatible (elle plante le web service), attention aux imports silencieux !

- L'envoi de mail : l'instruction Send ne fonctionne pas, il faut utiliser le meladx.exe pour faire un workflow serveur.

4.4.1 Les règles de développement

- Les éditions : instruction Report. Toujours l'utiliser avec un serveur d'édition.
  - Le paramètre superviseur SERIMP est à renseigner

- Interruption du programme par l'ouverture d'une fenêtre :

```
Call DIALWIN(W_OK,mess(113,188,1),"ACOPIE") From
  GESECRAN
```

  N'est pas compatible en mode web service

- L'instruction inpbox hors modélisation objet est à proscrire

4.4.2 Test avec l'applicatif
Les programmes wrapper

- Ils sont générés lors de la publication d'un objet ou d'un sous programme
  - ***WJ+«Nom d'objet»+«nom de variante»*** pour les objets sans publication des champs invisibles
  - ***WJ+«Nom d'objet»+«nom de variante»_I*** pour les objets avec publication des champs invisibles
  - ***WR+«Nom de publication»*** pour les sous-programmes

- Ils constituent une interface entre le développement X3 et le serveur de Web services. Ils gèrent
  - Les paramètres d'entête
  - Les messages
  - Le flux montant et descendant de données

- Les programmes wrapper sont appelés par le serveur de Web services dont le rôle est de sérialiser / désérialiser le flux SOAP de / vers les paramètres attendus.

4.4.2 Test avec l'applicatif
Paramètres des programmes wrapper

- Les paramètres sont normalisés en variables d'entête et flux de données. Les variables d'entête servent à contrôler les flux émis et reçus. Leur description est la suivante :
  - **WW_OK** **Integer** Statut de retour 1=OK , 0=KO
  - **WW_ZONE** **Character** Zone de sortie en cas d'erreur
  - **WW_STAT** **Integer** Nombre de messages
  - **WW_GRAVE** **Integer(1..50)** Gravité:1=Info, 2=Warning, 3=Bloquant
  - **WW_MESS** **Character(1..50)** Texte du message
  - **WW_ACTION** **Character** Code action à exécuter l'action
  - **WW_IDENT** **Character** Identifiant (Cle1~Cle2~Cle3…) ou clé début, ou critères de sélection rapide si liste
  - **WW_NB** **Integer** Nombre d'enregistrements à lire/lus
    WW_HORDAT **Character** Horodatage de l'élément
  - **WW_TAB** **Character** Identifiant de tableau
    - (futures actions SUPLIG et INSLIG)
  - **WW_PAR** **Character** Identifiant de lignes (idem WW_TAB)
  - **WW_TRACE** **Clbfile** Trace à renvoyer au serveur de Web services. Paramètres de mise en œuvre du debugger

4.4.2 Test avec l'applicatif
Programmes Wrapper

- La génération du programme wrapper est une des étapes de la génération du Web service. A l'intérieur du programme wrapper on trouve un programme de test qui permet de valider le fonctionnement de l'objet ou du sous-programme en mode Web service, en simulant une exécution, sans avoir à tester depuis l'application tierce.

- Lors de l'appel d'un Web service, le serveur de Web service transmet au programme wrapper l'horodatage de sa structure locale. Le serveur X3 vérifie si cet horodatage est conforme à celui stocké sur le serveur. Si oui il répond à la requête, si non il renvoie une erreur et le serveur de Web service envoie une requête technique pour récupérer la dernière version.

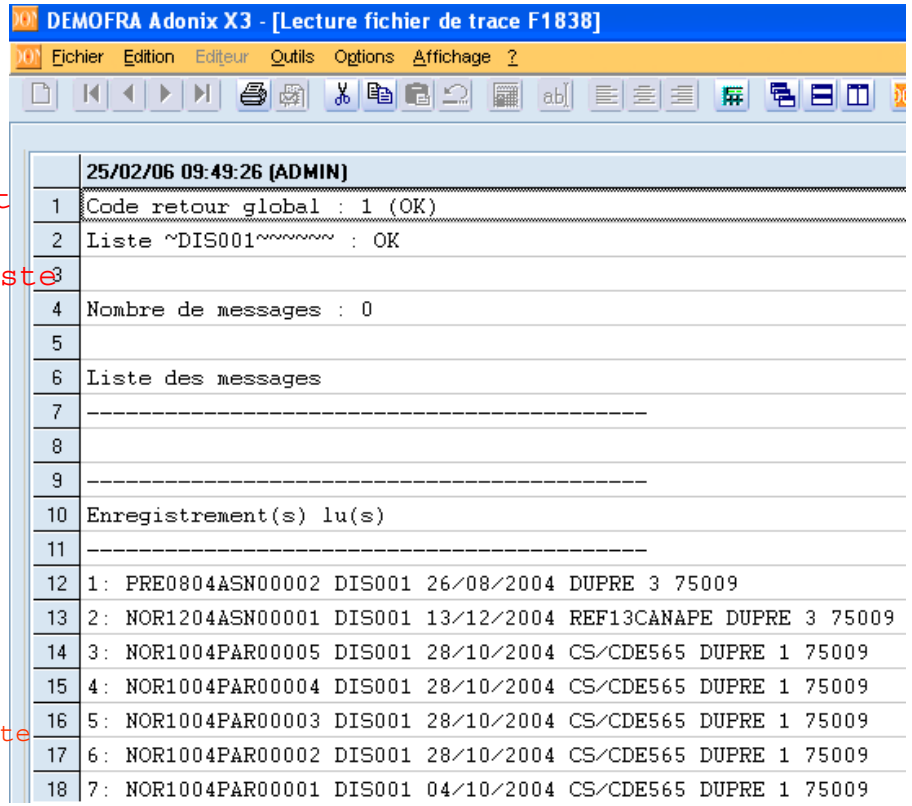
- Le mode opératoire consiste à développer un programme de test séparé du programme wrapper, car toute modification de celui-ci sera perdue lors de la republication de l'objet.

4.4.2 Test avec l'applicatif
Test liste gauche

```
################################
## Test liste gauche
################################
Call TEST_LISTE ( 10 )    # Nombre de lignes maximum
End

Subprog TEST_LISTE(NBLISTE)
Value Integer NBLISTE
Call OUVRE_TRACE("") From LECFIC
Gosub DEFLISTE From WJSOHSTD
WW_ACTION="LIST"          # Action liste gauche
WW_IDENT="~DIS001~~~~~~" # Critère:commande du client
 DIS001
Gosub WEBLISTE From WJSOHSTD # Déclenchement de la liste
Gosub ANALRESOBJ From WJSOHSTD # Analyse du résultat
For indice=1 To WW_NB
  If indice=1
    Call ECR_TRACE(mess(223,123,1),0) From GESECRAN
    Call ECR_TRACE("----------",0) From GESECRAN
  Endif
  Call ECR_TRACE(num$(indice)+":"
&  -WW_SEL1(indice)
&  -WW_SEL2(indice)
&  -WW_SEL3(indice)
&  -WW_SEL4(indice)
&  -WW_SEL5(indice)
&  -WW_SEL6(indice)
&  -WW_SEL7(indice),0) From GESECRAN # Affichage de la liste
Next indice
Call ECR_TRACE("----------------",0) From GESECRAN
Call FERME_TRACE From LECFIC
Call LEC_TRACE From LECFIC
End
```

Trace du résultat



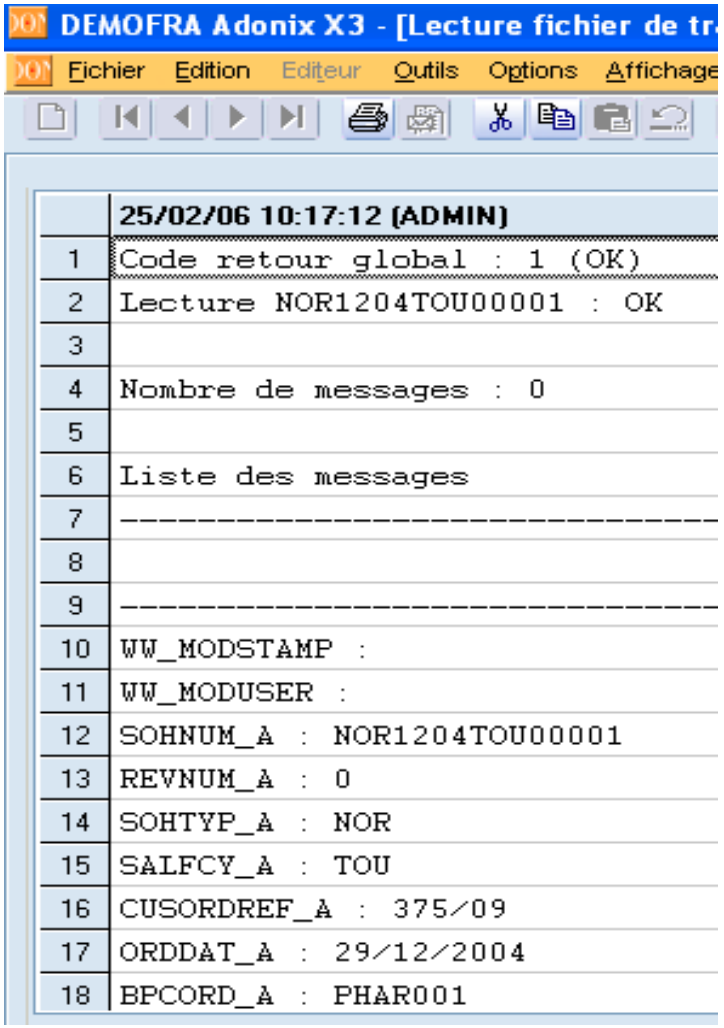
4.4.2 Test avec l'applicatif
Test lecture objet

- L'étiquette Gosub DETAILRES permet d'afficher dans une trace :
  - Le statut global
  - l'enregistrement qui a été lu ou écrit (tous les champs des blocs liste et la première ligne de chaque tableau)
  - La liste des messages

```
#########
## Test de lecture
#########
Call TEST
End

Subprog TEST
Local Char    TEXTE(20)
Local Integer I , J , K
Call OUVRE_TRACE("") From LECFIC
Gosub DEFVAR From WJSOHSTD : # Appel programme wrapper
WW_ACTION = "READ"  : # READ, CREATE, MODIFY, DELETE
WW_IDENT  = "NOR1204TOU00001"   : # Clé de l'objet Val1~Val2
Gosub WEBSERV From WJSOHSTD : # Simulation web service
Gosub ANALRESOBJ From WJSOHSTD : # Affichage du statut
Gosub DETAILRES From WJSOHSTD : # Affichage des champs
Call FERME_TRACE From LECFIC
Call LEC_TRACE From LECFIC
End
```

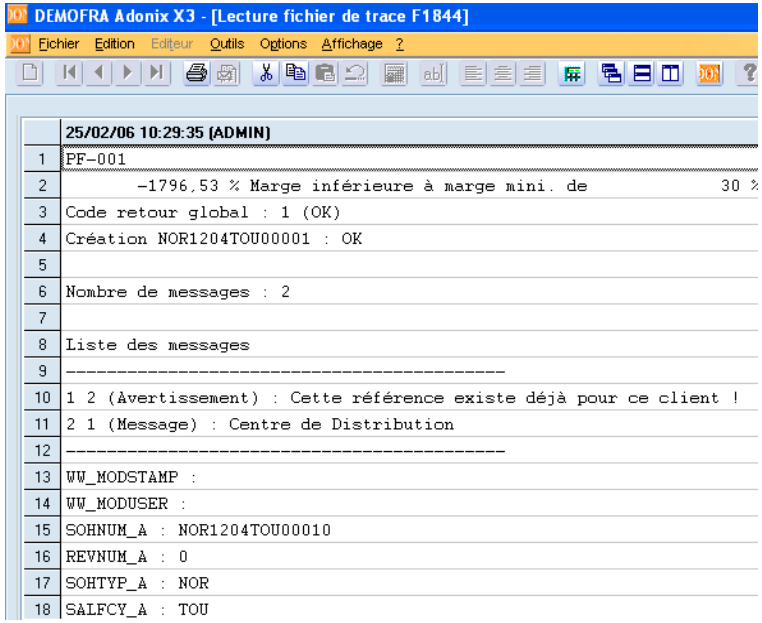
Trace du résultat →



	25/02/06 10:17:12 (ADMIN)
1	Code retour global : 1 (OK)
2	Lecture NOR1204TOU00001 : OK
3	
4	Nombre de messages : 0
5	
6	Liste des messages
7	----------
8	
9	----------
10	WW_MODSTAMP :
11	WW_MODUSER :
12	SOHNUM_A : NOR1204TOU00001
13	REVNUM_A : 0
14	SOHTYP_A : NOR
15	SALFCY_A : TOU
16	CUSORDREF_A : 375/09
17	ORDDAT_A : 29/12/2004
18	BPCORD_A : PHAR001

4.4.2 Test avec l'applicatif
Test création objet

Trace du résultat →




```
############
## Test de création
############
Call TEST
End

Subprog TEST
Local Char    TEXTE(20)
Local Integer I , J , K
Call OUVRE_TRACE("") From LECFIC
Gosub DEFVAR From WJSOHSTD
WW_ACTION = "READ" :# On commence par lire
WW_IDENT  = "NOR1204TOU00001" : # Clé de l'objet
 Val1~Val2
Gosub WEBSERV From WJSOHSTD : # Simulation lecture
WW_ACTION = "CREATE"       : # Maintenant on crée
Gosub WEBSERV From WJSOHSTD : # Création
Gosub ANALRESOBJ From WJSOHSTD : # Analyse du résultat
Gosub DETAILRES From WJSOHSTD : # Affichage des champs
Call FERME_TRACE From LECFIC
Call LEC_TRACE From LECFIC
End
```


4.4.2 Test avec l'applicatif
Test de sous programme

- Dans le programme wrapper un sous-programme de test est intégré.

- Ce sous-programme appelle une étiquette **INITWS** avant l'appel du sous-programme et une étiquette **RESULTWS** après l'appel

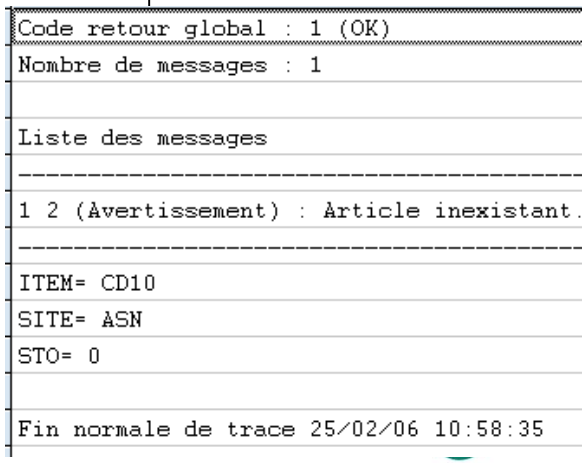
- Ceci permet à l'intérieur même du programme à publier de prévoir des séquences de test (elles ne seront appelées que par le programme de test intégré avec le wrapper) afin :
  - De charger des variables avant l'appel,
  - D'écrire des résultats dans la trace après l'appel
  - L'écriture des messages est gérée automatiquement par le programme de test.

4.4.2 Test avec l'applicatif
Test de sous programme

```
Subprog STOCK ( ITEM, STO, SITE )
Variable Char ITEM
Variable Decimal STO
Variable Char SITE
 Local File ITMMVT [ITV]
Local File ITMMASTER [ITM]
Read [F:ITM]ITM0=ITEM
If fstat
 Call ADDMESSWARN("Article inexistant.") From AWEB
Else
 STO=0
 For [ITV] Where ITMREF=ITEM and STOFCY=SITE
  STO+=[ITV]PHYSTO
 Next
Endif
End
 $INITWS : # Etiquette pour les tests
ITEM="CD10"
SITE="ASN"
STO=0
Return
 $RESULTWS : # Etiquette pour afficher le résulat
Call ECR_TRACE("ITEM="-ITEM,0) From GESECRAN
Call ECR_TRACE("SITE="-SITE,0) From GESECRAN
Call ECR_TRACE("STO="-num$(STO),0) From GESECRAN
Return
```

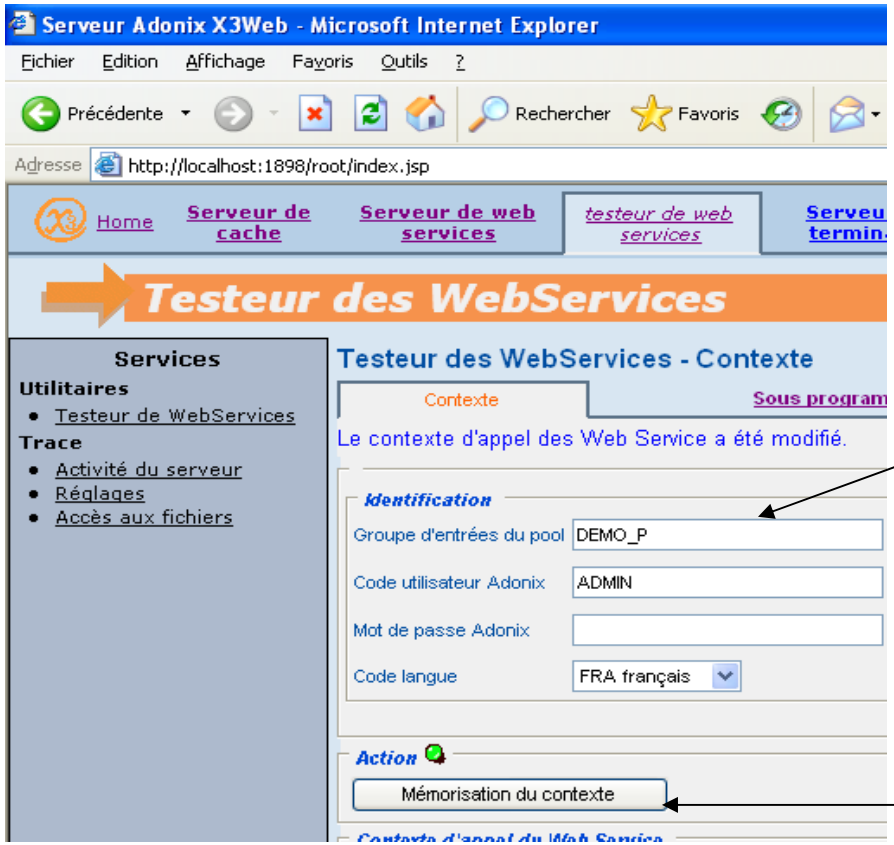
Trace du résultat

(Run WRSTOCK)



4.4.3 Test avec le serveur WEB

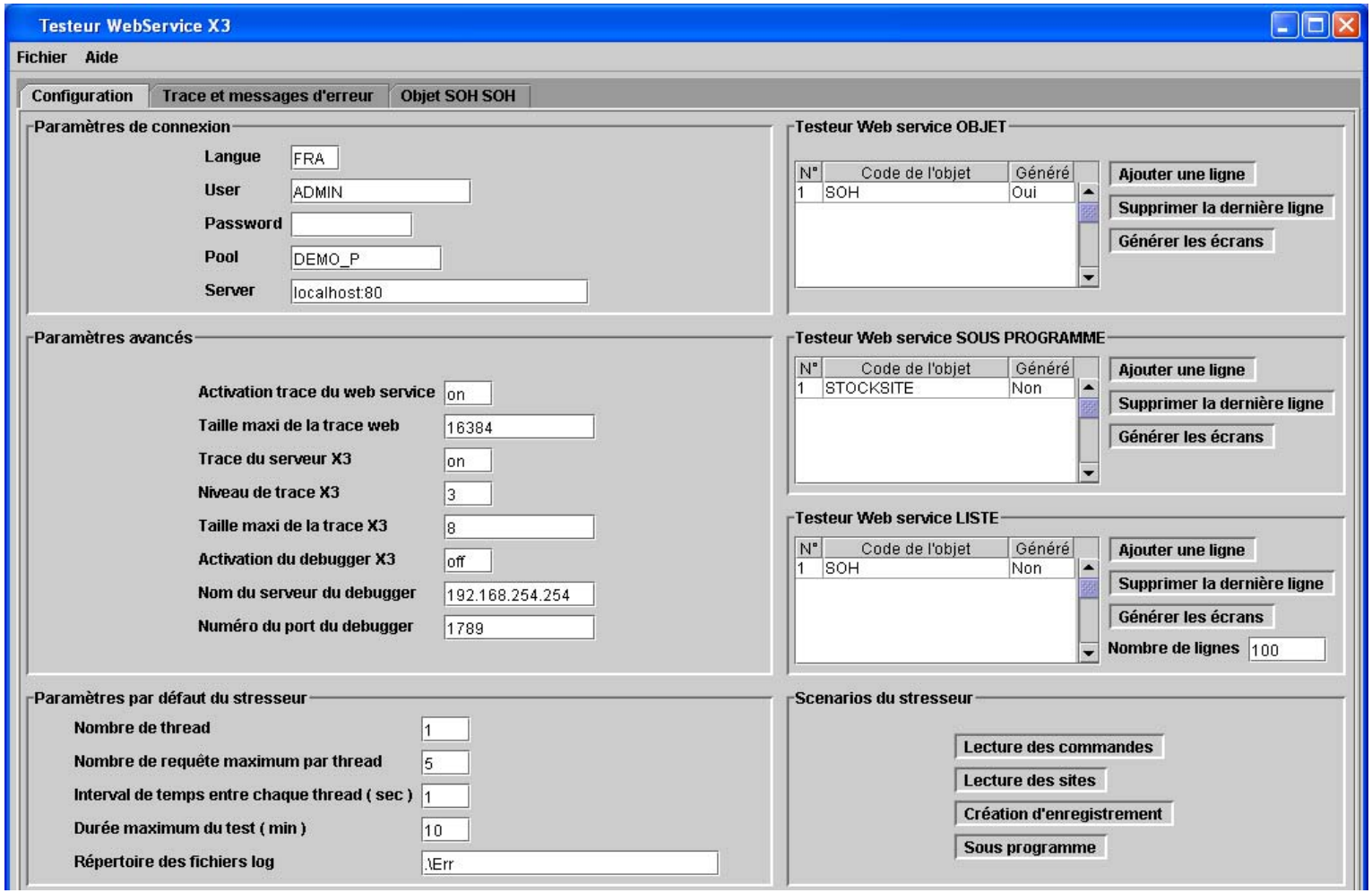
- Le serveur WEB inclus un testeur de web service, celui ci permet de valider le fonctionnement des objets modélisés dans X3.
- Il s'agit d'un client web service, il accessible depuis la page technique



- Saisie des paramètres de connexion
- Mémorisation du contexte

4.4.3 Test avec le serveur WEB

4.4.4 Le testeur JAVA



4.4.4 Le testeur JAVA

- Le testeur java est un outil de validation technologique muni d'une interface de contrôle et de saisie, il permet de s'affranchir dans un premier temps de la problématique de constitution des flux XML.
- Il est fourni à titre d'exemple et ne fait pas parti des livrables de la solution Sage X3

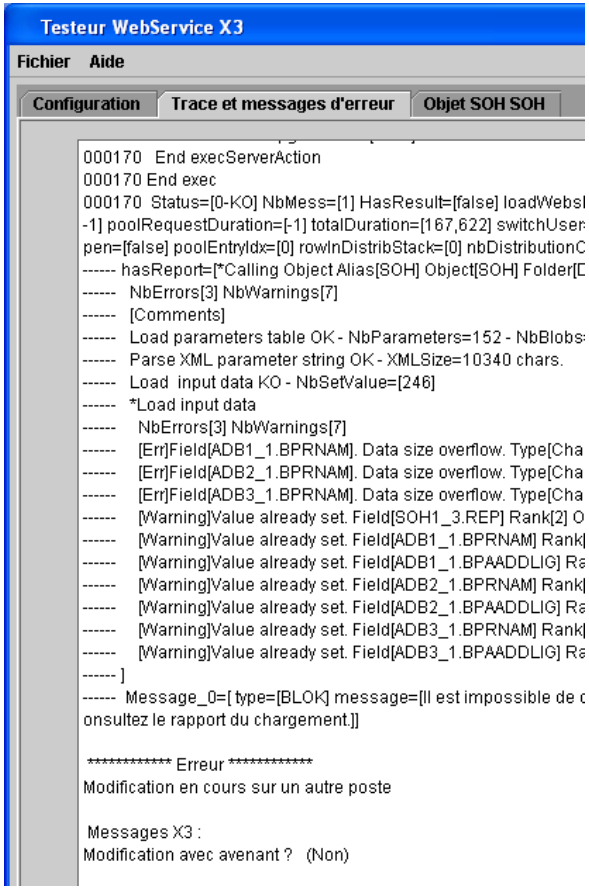
Cet outil,

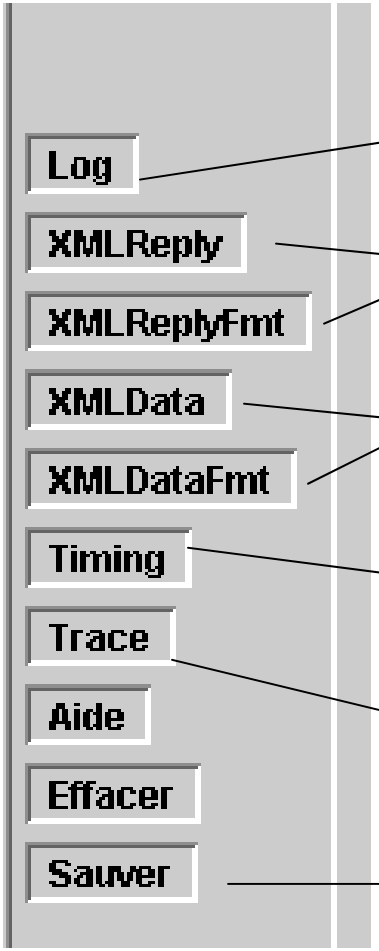
- Implémente les trois web service (Objet / liste / s/s programme)
- Affiche les traces d'exécution
- Affiche les flux XML de la réponse
- Donne les temps d'exécution
- Construit les flux XML automatiquement
- Implémente un outil de test de charge

Il utilise le fichier Def.Xml contenant les paramètres de base pour accéder aux WEB services X3

4.4.4 Le testeur JAVA

- L'onglet trace et message d'erreur donne des informations sur l'exécution du web service invoqué

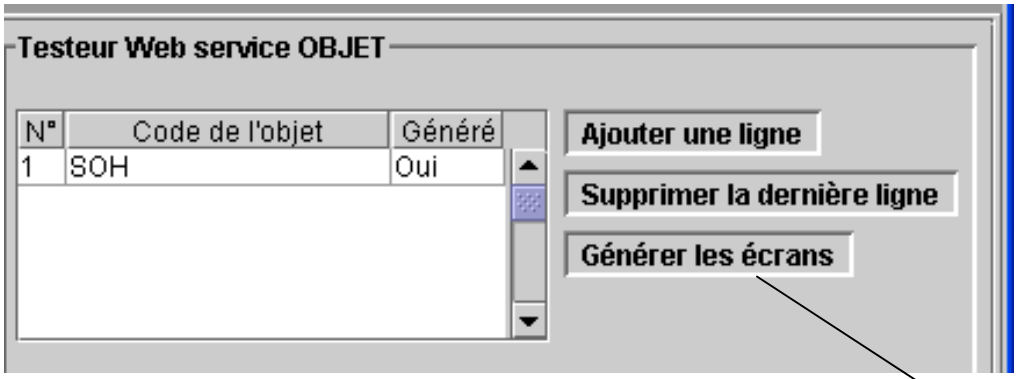




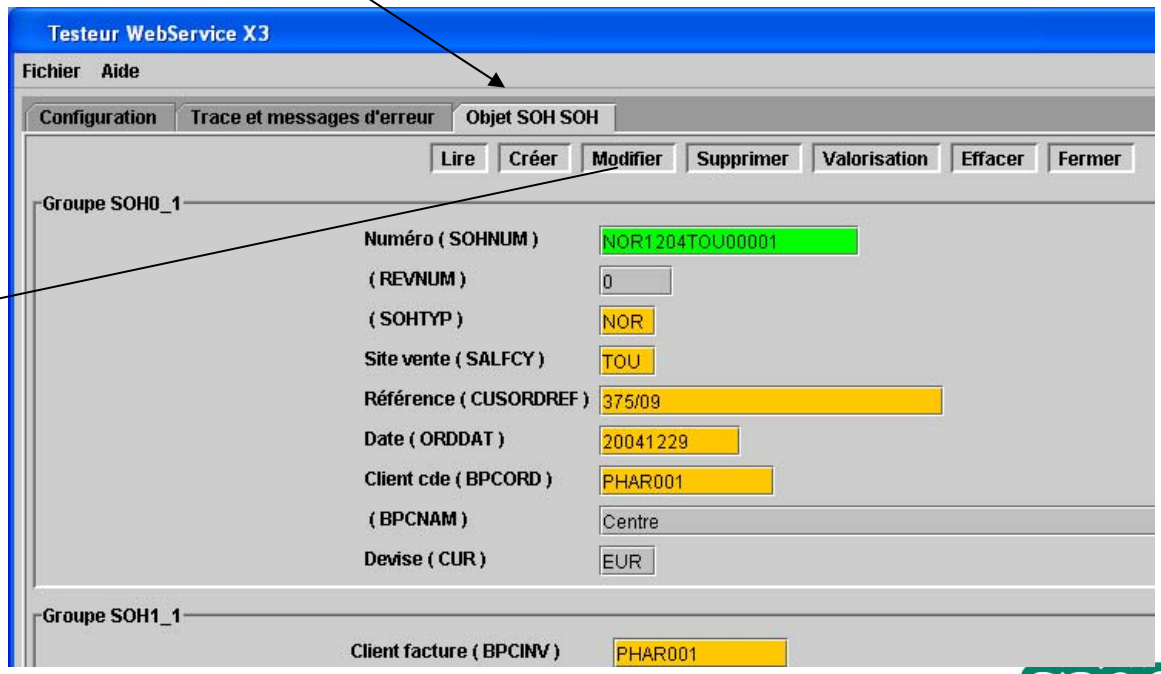
Dans cet onglet on peut :

- Afficher la trace du serveur web
- Afficher le flux renvoyé par le serveur
- Affiche les flux XML de la réponse
- Avoir des informations sur les temps
- Afficher la trace
- Sauver ces informations sur disque

4.4.4 Le testeur JAVA

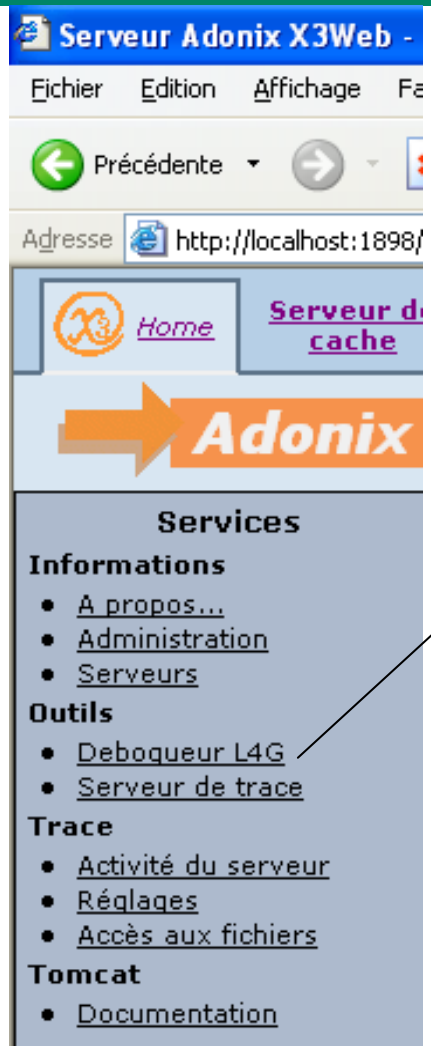


- La génération des écrans appelle le web service getdescription et ajoute un onglet dans l'interface

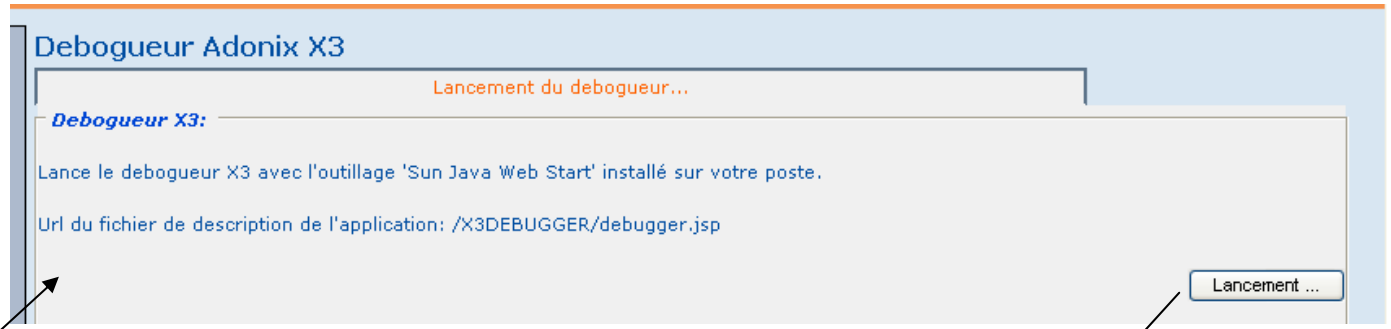


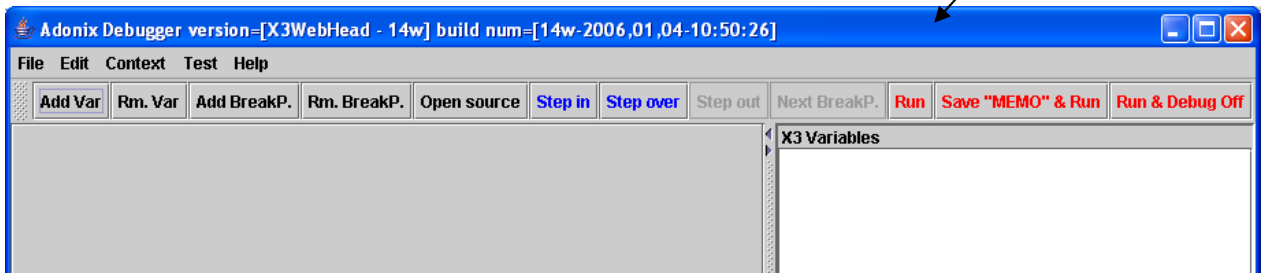
- L'interface permet d'invoquer les autres web services

4.4.5 Debugger les web service



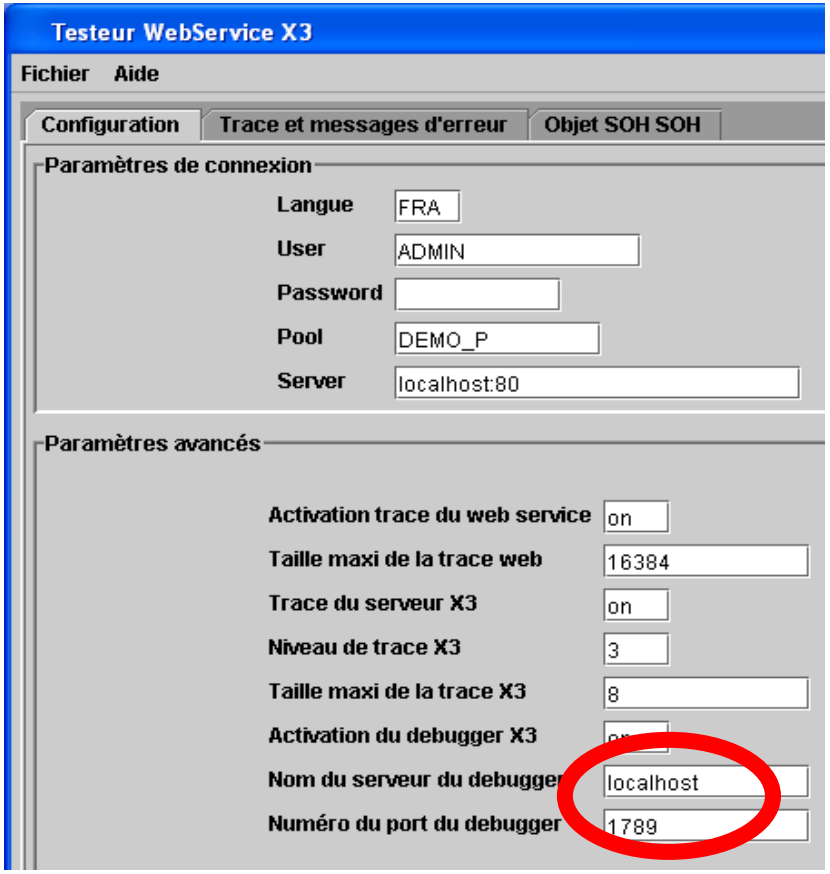
- Pour debugger en mode web service il faut utiliser le debugger java
- Il est a télécharger depuis le serveur web
- Puis à lancer

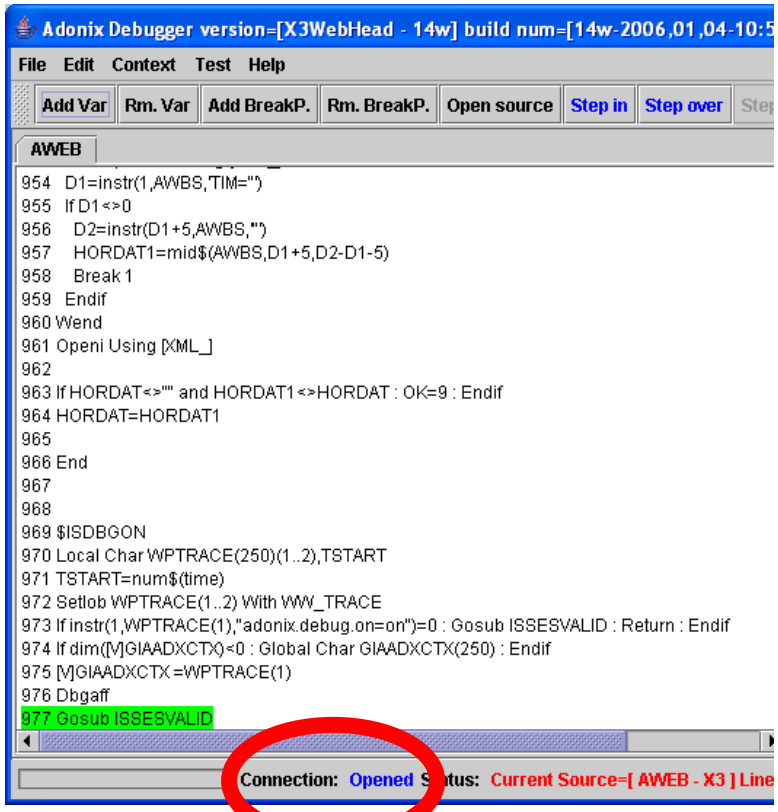




4.4.5 Debugger les web service

- Une fois le debugger lancé il faut invoquer un web service en précisant dans la chaine de connexion :
  - Le nom du serveur du debugger
  - Le numéro de port (défaut 1789)





- Le debuggeur s'arrête dans le traitement AWEB puis sur chacun des dbgaff ajoutés dans le code

4.5 Les tests de charge

- Une fois la mise au point terminée, il faut vérifier la performance globale du système web service. Cela implique d'avoir au départ des informations sur le nombre maximum de requête envoyée sur le serveur WEB X3. Il faut se concentrer sur les pics d'activité du site web par exemple.

- Il faut savoir qu'une création d 'enregistrements par le biais du web service objet est moins rapide qu'un import.

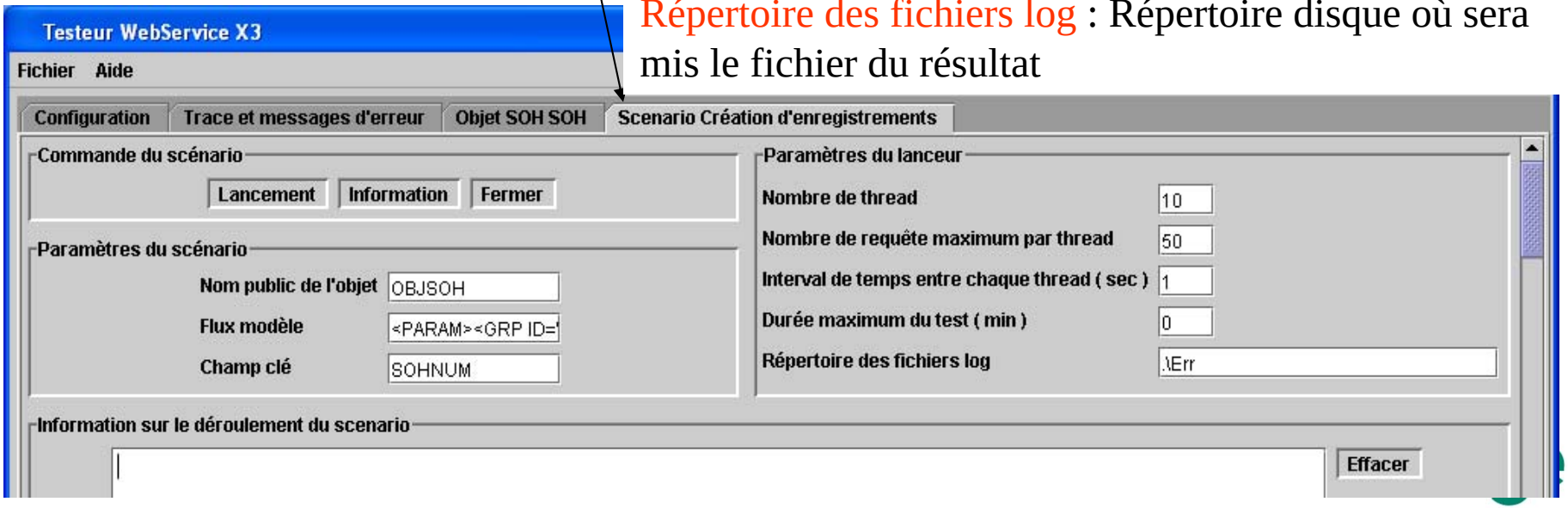
- Les tests de charge permettent aussi de déterminer le nombre de licence nécessaire pour avoir des temps de réponse acceptable.

- Ils permettent d'avoir une idée de la charge CPU et de la consommation mémoire sur le serveur web et sur le serveur X3. Le parsing des fichiers XML est consommateur de ressources.

- Les tests de charge peuvent être réalisés par le testeur JAVA qui permet de simuler une activité utilisateur en fonction d'un scénario donné et pour nombre simultané d'utilisateur paramétré.

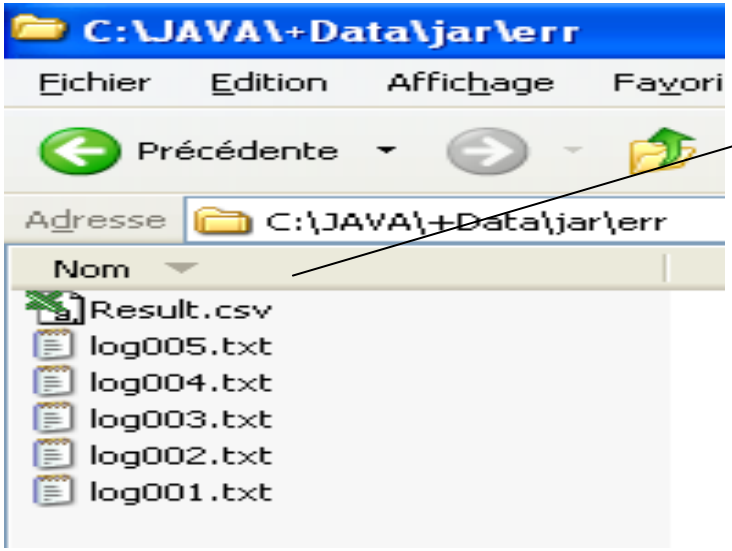
4.5 Les tests de charge

Nom public de l'objet : Nom de publication dans X3
Flux modèle : flux xml envoyé au serveur
Champ clé : Nom du champ clé de l'objet
Nombre de thread : Nombre d'utilisateurs simultanés
Nombre maxi de requête : Nombre de demande pour chaque utilisateur (envoyée séquentiellement)
Interval de temps : Temps d'attente entre chaque utilisateur supplémentaire
Durée minimum : Durée au delà de quelle le test se termine automatiquement
Répertoire des fichiers log : Répertoire disque où sera mis le fichier du résultat



4.5 Les tests de charge
Le résultat

- L'exécution du stresseur produit un fichier csv exploitable sous excel :



Microsoft Excel - Result.csv

Fichier Edition Affichage Insertion Format Outils Données Fenêtre ?

F20 =

	A	B	C	D	
1	Process	N°	Info	Error	T0
2	1	1	EXEC STOCKSITE	0	11408
3	1	2	EXEC STOCKSITE	0	11408
4	1	3	EXEC STOCKSITE	0	11408
5	2	1	EXEC STOCKSITE	0	11408
6	2	2	EXEC STOCKSITE	0	11408
7	2	3	EXEC STOCKSITE	0	11408
8	3	1	EXEC STOCKSITE	0	11408
9	3	2	EXEC STOCKSITE	0	11408
10	3	3	EXEC STOCKSITE	0	11408
11	4	1	EXEC STOCKSITE	0	11408
12	4	2	EXEC STOCKSITE	0	11408
13	4	3	EXEC STOCKSITE	0	11408
14	5	1	EXEC STOCKSITE	0	11408
15	5	2	EXEC STOCKSITE	0	11408
16	5	3	EXEC STOCKSITE	0	11408

Process : Numéro de l'utilisateur

N° : Numéro de la requête pour un utilisateur

Info : Indique l'objet du web service

Error : Si = 1 une erreur s'est produite

T0 : Time stamp de la demande

T1 : Time stamp de la réponse

T1-T0 : Temps d'attende

rowInDistribStack : Rang d'empilage dans la file d'attente à l'arrivée de la requête

nbDistributionCycle : Nombre de test effectué avant de trouver une connexion disponible

poolEntryIdx : N° de l'entrée dans le pool de connexion

poolDistribDuration : Temps de recherche d'une connexion libre

poolExecDuration : Temps d'exécution du traitement ADONIX

poolWaitDuration : Temps d'attente de la requête avant d'être en tête de la liste d'attente

poolRequestDuration : Temps total de la demande de traitement

Mise en œuvre des WEB service X3
Résumé

Pour utiliser les services WEB X3 il faut :

- Installer et configurer le serveur WEB X3
  - Rapide et simple à faire peu de paramètres à renseigner tout est automatique
- Développer en L4G les sous programmes
  - A faire uniquement si la tâche a réaliser n'est pas modélisée sous forme d'objet
- Publier les objets et les sous programmes concernés
  - Il faut décrire au préalable les sous programme dans le dictionnaire
  - C'est une opération très rapide à réaliser
- S'assurer de leur compatibilité dans ce mode d'utilisation
  - Sur l'objet des commandes de vente rien est à faire
  - Pour les autres objets, seul des tests peuvent donner une idée du travail à accomplir, notamment en ce qui concerne le spécifique

Mise en œuvre des WEB service X3
Les évolutions prévues pour la release V5.1

Les appels de Web service SOAP

Architecture :

- Les appels de Web service SOAP sont réalisés via le composant "**Sage X3 Java Server**"
- L'outillage d'appel de Web services SOAP est un "plugin" de ce composant

Le plugin d'appel des Web services SOAP :

- Il utilise le clients de Web Service Apache Axis2 en version 1.2 (http://ws.apache.org/axis2/)
- Ce client respecte les spécifications :
  - ✓"WS-I" (voir http://www.ws-i.org/ et http://fr.wikipedia.org/wiki/WS-I),
  - ✓"WS-Security" (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)
- Les Web services tiers pouvant être appelés sont donc ceux qui respectent les spécifications "WS-I Basic Profile" et "WS-Security"

Mise en œuvre des WEB service X3
Les évolutions prévues pour la release V5.1

Les appels de Web service SOAP

Depuis le L4G X3, le mode opératoire d'un appel est :

- Ouverture d'une session avec un serveur "Sage 3 Java Server" (cf. Opadxd)
- Un appel pour obtenir une référence sur une instance du stub du Web service tiers à appeler
- Un ou des appels pour créer le ou les "java beans" (cf. structures) correspondant au(x) paramètre(s) nécessaire à l'action du Web service tiers
- Appel de la méthode du stub correspondant à l'action du Web service tiers à appeler
- Un ou des appels pour lire le ou les "java beans" (cf. structures) correspondant au(x) valeurs(s) retournées du Web service tiers (ex: un arbre...) .

Mise en œuvre des WEB service X3
Les évolutions prévues pour la release V5.1

Les appels de Web service REST

Architecture :

- Les appels de Web service SOAP sont réalisés via le composant **"Sage X3 Java Server"**
- L'outillage d'appel de Web services REST est un "plugin" de ce composant

Le plugin d'appel des Web services REST :

- Il utilise le clients HTTP produit par le sous projet «Apache Jakarta HttpComponents » http://jakarta.apache.org/httpcomponents/index.html
  - ✓Standards based, pure Java, implementation of HTTP versions 1.0 and 1.1
  - ✓Full implementation of all HTTP methods (GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE) in an extensible OO framework.
  - ✓Supports encryption with HTTPS (HTTP over SSL) protocol.
  - ✓Transparent connections through HTTP proxies.
  - ✓Tunneled HTTPS connections through HTTP proxies, via the CONNECT method.

Mise en œuvre des WEB service X3
Les évolutions prévues pour la release V5.1

Les appels de Web service REST

Depuis le L4G X3, le mode opératoire d'un appel est :

- Ouverture d'une session avec un serveur "Sage 3 Java Server"
- (nouvelle instruction Opadxd)
- Un appel pour obtenir une référence sur une instance du client HTTP
- Un ou des appels pour configurer cette instance (mime type attendu, ...)
- Appel de la méthode GET, POST,PUT,DELETE de l'instance
- Un ou des appels pour lire les données (xml, json, etc...) délivrées par le Web Service tiers.

Travaux pratiques

Exercice 1

(voir annexe)

- Publication et tests sur les web services
  - Objet
  - Liste
  - Sous programmes

- Test de ces objets pour chacune des méthodes

Formation Web Services Adonix

5. Intégration dans l'application cliente

   5.1. Flux XML reçu

   5.2. Flux XML émis

   5.3. Application cliente en JAVA

   5.4. Application cliente DOTNET

5 Intégration dans l'application cliente

- L'objectif est d'invoquer les web service X3 depuis une application tierce.
- Le standard web service comporte une norme qui décrit le service WEB, il se présente sous la forme d'un fichier XML. Ce fichier (*.wsdl) décrit les méthodes et les propriétés fournies par le service.
- Mais dans le contexte où l'application serveur est un ERP qui n'est pas figé dans sa structure il n'est pas possible de fournir un service web immuable.
- Pour ne pas avoir à modifier les applications clientes parce qu'on a ajouté un champ dans un écran, nous proposons des services web génériques où seules les méthodes sont fixes.
- La partie variante est constituée des paramètres à transmettre à ces méthodes. Ceux – ci sont transmis via un flux XML qui est a coder et à décoder dans les applications clientes.
- De nombreuses librairies sont disponibles dans différents langages pour effectuer ce travail.

5 Intégration dans l'application cliente

- Le serveur web 14W31 incorpore plusieurs versions des web services X3, ceux-ci sont présentés dans un tableau sur le serveur web :

- Le tableau croise la version du web service avec les plate formes client qui vont les utiliser
- Java : les exemples sont fournis sous la forme d'un workspace Eclipse 3.1
- Dot NET : les exemples sont fournis sous la forme d'une solution visual studio 2003 ou 2005

- Chaque version à ses avantages et inconvénients ...

5 Intégration dans l'application cliente

Les WebServices	JAVA	.NET
CAdxObjectXml (wsdl), **CAdxObjectListXml** (wsdl), **CAdxSubProgramXml** (wsdl) Ancienne version web service 14W20, 3 webs services distincts, la compatibilité ascendante est assurée. Il nécessite une connaissance de SOAP pour ajouter le header manquant.	Classe générées à partir des wsdl (ce qui évite de les générer soi-même) Les classes stub Les sources des classes stubs Exemple en Java livré sous la forme d'un projet Eclipse (environnement de développement) Le workspace Eclipse 3.1	DLL pour utiliser ce web service dans dot Net et son source L'assembly dans un CAB signée. L'assembly utilisable dans un projet Visual Studio Exemple pour utiliser ces web services en .NET La solution Visual Studio 2003
CAdxWebServiceXmlCC (wsdl), Nouvelle version du web service, le header à été remplacé, plus facile à utiliser. Il n'y a plus q'un WSDL	Exemple en Java livré sous la forme d'un projet Eclipse, les classes générées sont inclues dans le projet. Le workspace Eclipse 3.1	Exemple .NET 2005 pour utiliser directement ce web service donc sans la DLL qui n'est plus utile ici. On est plus proche du standard web service. La solution Visual Studio 2005
CAdxWebServiceDomCC (wsdl), Identique au web service précédent mais les informations sont récupérées sous la forme d'un arbre XML directement en mémoire, plus de parsing à réaliser	En java ce web service ne peut pas être utilisé dans une application cliente. Problème de compatibilité ...	Exemple .NET pour utiliser ce web sercice, fourni sous la forme d'une solution .net 2005. La solution Visual Studio 2005

5.1 Flux XML reçu

- **On obtient un flux résultat suite à :**
  - Lecture d'un objet
  - Modification ou création d'un objet
  - Suppression d'un objet
  - Exécution d'une option
  - Exécution d'une liste gauche
  - Exécution d'un sous programme

- **Le flux est compris dans un nœud RESULT**

```
<RESULT>
</RESULT>
```


5.1 Flux XML reçu

- **Dans le flux on peut recevoir :**
  - **Des groupes de champs (bloc liste ou paramètre de sous programme)**

```
<GRP
ID="ITM0_1">
</GRP>
```

  - **Des tableaux de champs (bloc tableau) size donne le nb de lignes**

```
<TAB DIM="4" ID="ITM5_2" SIZE="4">
</TAB>
```

  - **Les tableaux contiennent des lignes, num donne le numéro de la ligne**

```
<LIN NUM="3">
</LIN>
```


5.1 Flux XML reçu

- **Dans les nœuds GRP et LIN on peut avoir :**
  - **Des champs simples**

```
<FLD NAME="ITMREF" TYPE="Char">CD100 </FLD>
```

  - **Des champs de type menu local, MENULAB contient l'intitulé du menu**

```
<FLD MENULAB="Actif" MENULOCAL="246" NAME="ITMSTA"
TYPE="Integer">1</FLD>
```

- **Dans les noeuds GRP on peut avoir aussi :**
  - **Des champs dimensionnés :**

```
<LST NAME="TSICOD" SIZE="1" TYPE="Char">
        <ITM>JOU</ITM>  (1ère occurrence)
        <ITM></ITM>     (2nd occurrence)
        <ITM>ECR</ITM>  (3ème occurrence)
</LST>
```


5.1 Flux XML reçu

- **Pour un sous programme on a :**
  - Des champs simples si la dimension = 1
  - Un ou plusieurs tableaux regroupés en cardinalité homogène

- **Pour une liste gauche on a un seul tableau**

5.2 Flux XML envoyé

- **On doit transmettre un flux pour**
  - Créer un objet
  - Modifier un objet
  - Exécuter un sous programme
- **Le flux est compris dans un nœud PARAM**

  <PARAM>

  </PARAM>
- **Le flux à transmettre est identique au flux reçu, mais il peut être plus souple :**
  - Si le champ peut être identifié sans ambiguïté, il n'a pas besoin d'être dans un nœud groupe
  - Les types de données ne sont pas à préciser
  - Les intitulés des menus locaux sont inutiles
  - Seuls les champs modifiés sont à transmettre
  - Indiquer le nombre de lignes des tableaux est facultatif
  - Indiquer pour chaque ligne d'un tableau son numéro est facultatif

5.3 Application cliente en JAVA
5.3.1 Préparation de l'environnement

Les fichiers WSDL

- Il existe un standard de description des web services, la description est stockée dans un fichier WSDL (Web Service Description Langage)
- Ce fichier donne des informations sur les méthodes proposées par le service web ainsi qu'une description des paramètres à passer pour les utiliser
- Sur le serveur WEB vous trouverez 3 fichiers WSDL aux URL suivantes :
  - Version 14W20 :
    - ✓ http://webserver/adxwsvc/services/CAdxSubProgramXml?wsdl
    - ✓ http://webserver/adxwsvc/services/CAdxObjetctXml?wsdl
    - ✓ http://webserver/adxwsvc/services/CAdxObjectListXml?wsdl
  - A partir de la version 14W31
    - ✓ http://webserver/adxwsvc/services/CAdxWebServiceXmlCC?wsdl
- Dans une application cliente java ou .NET ces fichiers peuvent être directement utilisés pour générer le code java permettant l'appel aux web services respectifs

5.3 Application cliente en JAVA

Préparation de l'environnement

Solution 1 : intégration du WSDL et génération des classes stubs

Serveur de web service

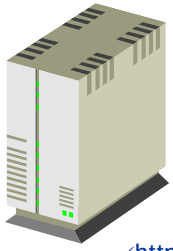


Plate forme de développement Java

Site de téléchargement

Récupération du fichier WSDL

✓http://webserver/adxwsvc/services/CAdxWebServiceXmlCC?wsdl

Téléchargement des packages ANT et AXIS

Génération des classes stubs

5.3 Application cliente en JAVA
Préparation de l'environnement

Solution 2: téléchargement des classes stubs depuis le serveur WEB

Serveur de web service

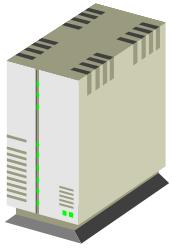


Plate forme de développement Java

Site de téléchargement

Récupération des classes stubs

Téléchargement des packages AXIS

5.3 Application cliente en JAVA

Préparation de l'environnement

- L'objectif est de constituer un flux SOAP qui soit conforme à ce que le serveur web attend, c'est à dire conforme à la description donnée dans le fichier wsdl. Les étapes pour y parvenir sont les suivantes :
  1. Chaque méthode appelée doit contenir un contexte d'appel, c'est l'un des paramètres de la méthode, il contient les informations suivantes :
     - ✓ Code langue X3
     - ✓ Code utilisateur X3
     - ✓ Mot de passe X3
     - ✓ Pool de connexion : il s'agit de l'alias donné dans la console
     - ✓ Serveur web : nom et port (80 ou 1898 par défaut) du serveur web X3
     - ✓ Chaine de configuration : indique les différents niveaux de trace pour la mise au point
  2. Les méthodes appelée nécessitent des paramètres supplémentaires. Ceux-ci sont regroupés dans un paramètre unique qui contient en réalité l'ensemble des paramètres d'un sous programme ou l'ensemble des champs d'un objet.
     - ✓ Ce paramètre est à construire manuellement dans l'application cliente, il doit se présenter sous la forme d'un flux XML.
  3. Une fois la méthode exécutée, celle – ci retourne des paramètres, ceux – ci sont regroupés en un seul.
     - ✓ Ce paramètre est à décoder, il est présenté sous la forme d'un flux XML qu'il faudra parser pour récupérer les informations

5.3 Application cliente en JAVA

Import des packages

```
/* Packages utilisés pour parser les flux XML */
import java.io.StringReader;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.xml.sax.InputSource;
/* Packages utilisés pour manipuler le flux xml */
import org.w3c.dom.Document;
import org.w3c.dom.NamedNodeMap;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;
/* Classes générées pour appeler le web service */
import com.adonix.wsvc.stubs.CAdxWebServiceXmlCCServiceLocator;
import com.adonix.wsvc.stubs.CAdxWebServiceXmlCC;
import com.adonix.wsvc.stubs.CAdxResultXml;// Classe générée contenant la réponse du serveur de web service
import com.adonix.wsvc.stubs.CAdxMessage; // Classe générée pour lire les messages retournés par l'applicatif
```


5.3 Application cliente en JAVA
La requête de configuration

- Il s'agit d'une chaîne de caractères qui fait partie du header à transmettre dans le flux SOAP, chaque paramètre est séparé par « & »

- Elle est constituée des éléments suivants :
  - adxwss.trace.on=(on/off) : active la trace du serveur WEB
  - adxwss.trace.size=16384 : taille maximum de la trace (ne pas changer)
  - adonix.trace.on=(on/off) : active la trace du serveur X3
  - adonix.trace.level=(1/2/3) : niveau de trace (paramètre en entrée, sortie, E/S)
  - adonix.trace.size=8 : taille maximum de la trace (taille du clob, ne pas changer)
  - adonix.debug.on=(on/off) : activation du debugger
  - adonix.debug.host=localhost : machine où est lancé le debugger
  - adonix.debug.port=1789 : port du debugger

5.3 Application cliente en JAVA
La requête de configuration

- **Deux façons de procéder**
  - Fabrication d'une chaîne de caractères en dur :

    String RequestConfigDebug = "
    adxwss.trace.on=on&adxwss.trace.size=16384&adonix.trace.on=on&adonix.trace.level=3&adonix.trace.size=8"; ou si pas de debug du tout :

    String RequestConfigDebug = " adxwss.trace.on=off&adonix.trace.on=off";

  - La reconstruire à chaque appel pour activer la trace ou pas

5.3 Application cliente en JAVA

Constitution du contexte d'appel

- **L'objectif est de construire un paramètre qui sera ajouté aux méthodes appelées. Ce paramètre est une classe (CAdxCallingContext) doit contenir les champs suivants :**
  - **Code langue,**
  - **code utilisateur,**
  - **mot de passe,**
  - **pool de connexion**
  - **et requête de configuration**

  Code java correspondant :

```
CAdxCallContext CallContext;
CallContext = new
  CAdxCallContext(Lan,User,Pwd,Pool,RequestConfig);
```

  - **Le CAdxCallContext est une des classes générées, elle stocke le contexte d'appel**

5.3 Application cliente en JAVA

Constitution du flux XML

- **Deux façons de procéder pour constituer le flux**
- **Exemple avec un flux pour demander le stock d'un article :**
  - **Fabrication d'une chaîne de caractères « en dur »**

```
String Parametre="<PARAM><GRP ID=\"G1\"><FLD
    NAME=\"ITEM\">"+Item+"</FLD></GRP></PARAM>";
```

Partie fixe

Partie variable

    - ✓ **Avantage : c'est simple à faire mais pas très souple**
    - ✓ **Inconvénient : le code sera a réécrire pour chaque appel à des objets ou sous programmes différents.**
  - **Construction manuelle d'un flux XML à l'aide des librairies DOM et parsing du flux pour obtenir une chaîne de caractères.**
    - ✓ **Avantage : une fois le code écrit il peut être réutilisé pour chaque flux à construire**
    - ✓ **Inconvénient : Plus complexe à mettre en œuvre (parsing du flux XML)**

5.3 Application cliente en JAVA
Instanciation du web service

- **L'objectif est de déclarer et de préparer le web service en vue de son exécution, pour cela deux étapes sont nécessaires :**
  - Déclarer un «locator » (url de destination)
  - Instancier le web service (utilisation des classes stubs générées)
- **Exemple avec un sous programme :**

```
private CAdxWebServiceXmlCCServiceLocator ServiceLocator;//URL de connexion
private CAdxWebServiceXmlCC Service; //Le sous programme X3

ServiceLocator = new CAdxWebServiceXmlCCServiceLocator();
// On indique l'URL du web service
ServiceLocator.setCAdxWebServiceXmlCCEndpointAddress(
 "http://localhost:1898/adxwsvc/services/CAdxWebServiceXmlCC");
// Instanciation de la classe stub contenant les fonctions d'appel webservice
Service = ServiceLocator.getCAdxWebServiceXmlCC();
```

- **Les classes Cadx... sont les classes générées (soit téléchargées sur le serveur web, soit générées depuis le WSDL)**

5.3 Application cliente en JAVA

Appel du web service

- **Appel du web service : L'objectif est d'invoquer le web service, cela revient à envoyer une requête HTTP contenant le flux SOAP à destination du serveur web X3**

  - **Exemple avec d'appel d'un sous programme (STOCK) :**

Déclaration de la classe de retour pour stocker le résultat :

```
private CAdxResultXml X3Reply; // Réponse retournée par le serveur de web/s
```

Appel du web service (la requête http part à ce moment là) :

```
X3Reply = Service.run(CallContext,"STOCK", XML);
```


5.3 Application cliente en JAVA
Décodage du résultat

- **Lors du retour du web service le résultat est stocké dans une classe CadxResultXml qui permet de :**
  - **Récupérer le flux de la réponse (c'est ici que x3 à mis le stock de notre article)**

```
String Result = X3Reply.getResultXml();
```

  - **Récupérer le statut (tout c'est bien passé ?)**

```
Int Statut = X3Reply.getStatus();
```

  - **Récupérer les messages (on récupère ici les messages applicatifs)**

```
CAdxMessage[] Messages = X3Reply.getMessages();
for ( int i = 0; i<Messages.length; i++ )
  System.out.println("Message "+i+"
 "+Messages[i].getMessage());
```

  - **Récupérer une structure avec les temps et les traces (très utile lors du débugage)**

```
CadxTechicalInfo = X3Reply.getTechnicalInfos();
```

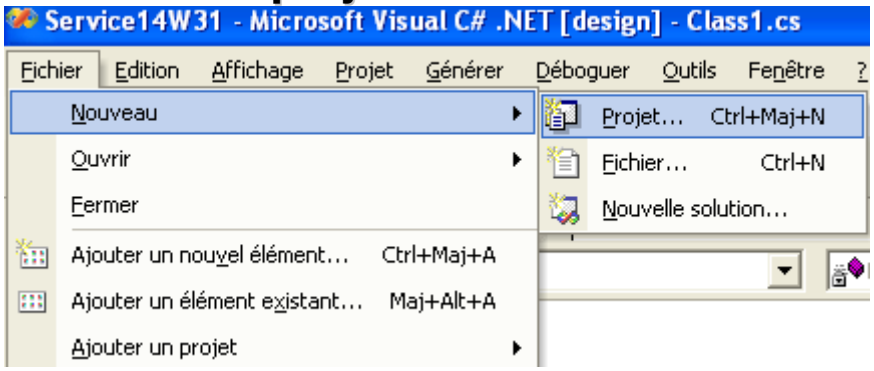

5.4 Application cliente en .NET

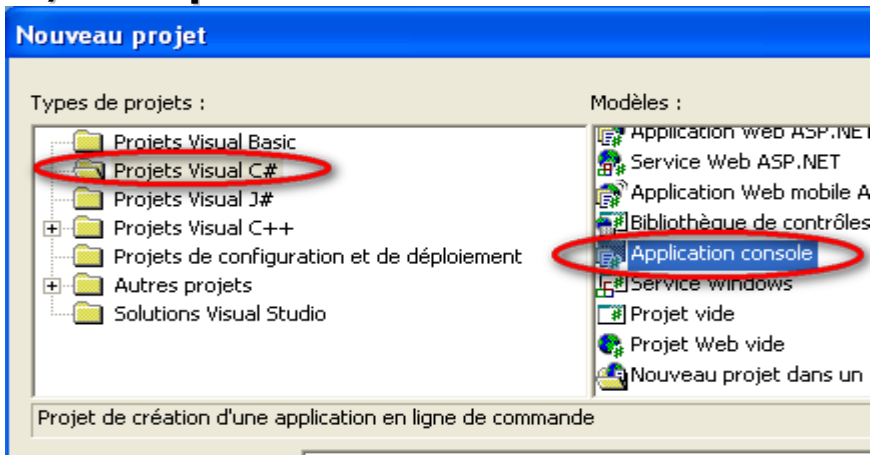
Préparation de l'environnement

- Tout comme java il y a une étape d'intégration de fichier WSDL, appelé référence web en environnement dot NET.
- Les soucis d'interopérabilité entre le serveur axis et le client dot NET nous imposaient en 14W25, l'utilisation d'une dll qui contenait déjà les classes stubs packagées pour être directement utilisable dans un projet dot NET.
- Depuis la version 14W31 il est possible d'intégrer dans un projet visual studio le fichier WSDL.
- Lors de son intégration les classes se génèrent automatiquement et sont directement utilisables.

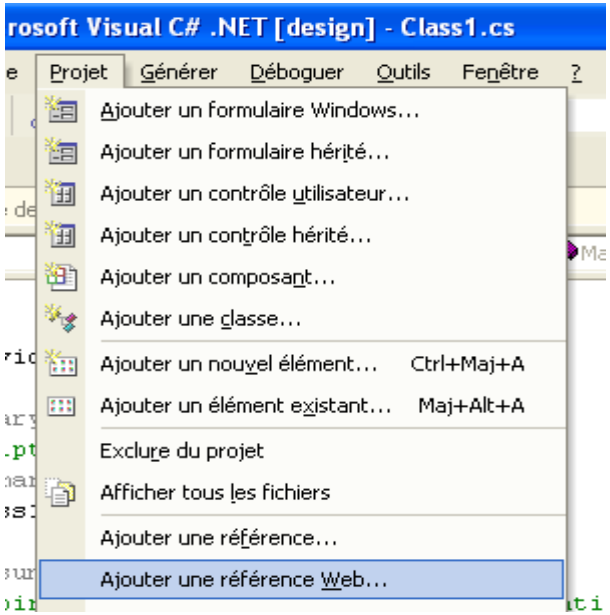
5.4 Application cliente en .NET
Préparation de l'environnement

- **Nouveau projet visual studio 2003/2005, exemple avec une console en C# …**



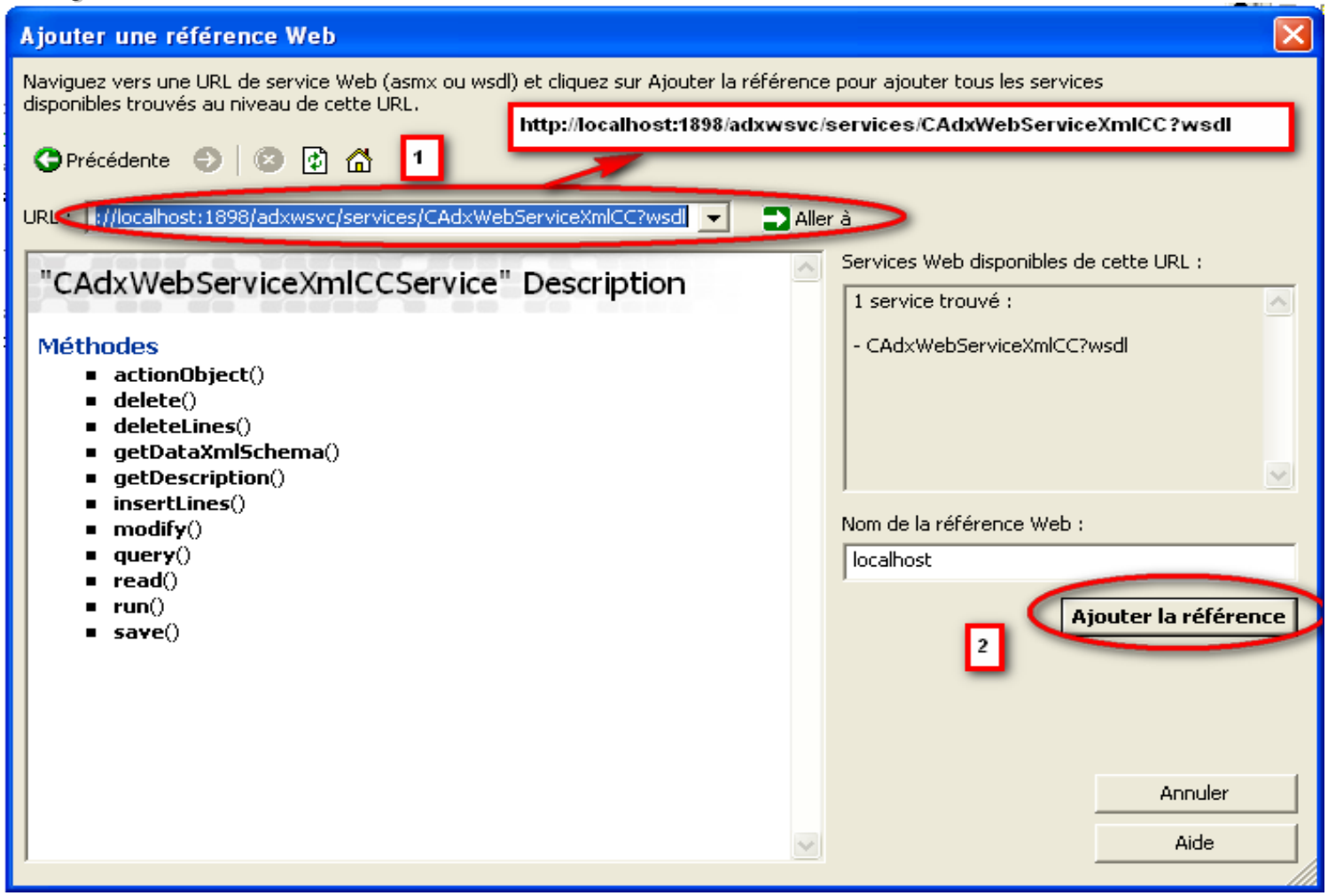


- **Puis intégration d'une référence WEB …**



5.4 Application cliente en .NET
Préparation de l'environnement

- **Ajout du WSDL :**



5.4 Application cliente en .NET
Le développement

- **Exemple commenté en C# :**
  - **En rouge les classes stub (générées)**
  - **En bleu les variables**
  - **En vert les commentaires**

1. **Déclaration des name space utilisés**

```
/* Classe de base */
using System;
/* Classe des web service X3 */
using Service14W31.localhost; // Le nom dépend du nom de la référence web
```

2. **Déclaration des variables utilisées**

```
CAdxWebServiceXmlCCService Subprog = new CAdxWebServiceXmlCCService(); // Le web service
CAdxCallContext Context = new CAdxCallContext(); // Le contexte d'appel
CAdxResultXml Resultat; // Le résultat de l'appel
```


5.4 Application cliente en .NET
Le développement

3. Alimentation du contexte

```
Context.codeLang="FRA";          // Code langue X3
Context.codeUser="ADMIN";        // Code utilisateur X3
Context.password="";             // Mot de passe”
Context.poolAlias="FORM";        // Pool de connexion
Context.RequestConfig=" adxwss.trace.on=on"; // Requête de configuration
```

4. Affectation de l’URL de la requête HTTP

```
Subprog.Url = "http://localhost:1898/adxwsvc/services/CAdxWebServiceXmlCC";
```

5. Appel de la méthode webservice et récupération du résultat

```
Resultat = Subprog.run(Context, "STOCKSITE",
            "<PARAM>
                    <GRP ID=\"G1\">
                            <FLD NAME=\"ITEM\">ECR01</FLD>
                            <FLD NAME=\"SITE\">ASN</FLD>
                    </GRP>
            </PARAM>");
```


5.4 Application cliente en .NET
Le développement

6. Récupération des messages d'erreur

```
Console.WriteLine("statut="+Resultat.status); //  Statut général de retour 0=Ok 1 = KO
for (int i=0; i<Resultat.messages.Length; i++) // Boucle pour récupérer les messages
{
 if (Resultat.messages[i].type.Equals("1")) Console.WriteLine("INFORMATION : ");
 if (Resultat.messages[i].type.Equals("2")) Console.WriteLine("AVERTISSEMENT : ");
 if (Resultat.messages[i].type.Equals("3")) Console.WriteLine("ERREUR : ");
 Console.WriteLine(Resultat.messages[i].message);
}
```

7. Récupération de la trace

```
Console.WriteLine("trace="+Resultat.technicalInfos.traceRequest);
```

